

AI Mindmeld: A Latent Interconnect Protocol

Siavash Golkar^{1,2} Jake Kovalic^{1,3} Shirley Ho^{1,2,4,5,6}

¹Polymathic AI

²Center for Data Science, New York University

³Department of Applied Physics, Yale University

⁴Center for Computational Astrophysics, Flatiron Institute

⁵Department of Astrophysical Sciences, Princeton University

⁶Department of Physics, New York University

September 2026

Abstract

Large language models (LLMs) are powerful tools for science, and tool calling enables them to act as autonomous agents that consult specialist models, run calculations and carry out experiments. To guide an agent’s reasoning, however, specialist information must enter the LLM’s context. Current-generation LLMs generally require that information to be translated into text, images, audio or video—a severe bottleneck between models that natively process rich, high-dimensional representations. We advocate direct representation exchange: a mindmeld through which an LLM could receive many properties together, including information it would not think to request. Such an interface should be extensible, making new specialists easy to add; flexible, allowing their information to support changing questions and tasks; and compositional, enabling independently connected specialists to work together without joint training.

We introduce the Latent Interconnect Protocol (LIP) to support this exchange. Its connectors are learned mappings that bring specialist representations directly into the LLM’s internal states. Keeping the language model frozen and only training the connectors makes this approach practical as LLMs grow and evolve. Our demonstration connects three protein and three small molecule specialists to three LLMs, Gemma 3 (4B and 12B parameters) and Kimi K2.6 (1T total, 32B active parameters) [53], while keeping all pretrained models frozen. Independently trained connectors provide useful information without changing the LLM or existing connectors, while supporting both familiar and new question types about unseen objects. We also demonstrate zero-shot transfer to cross-domain tasks without joint training. Task performance and compositionality generally improve with LLM capability, with particularly strong gains in zero shot task transfer. The demonstration also exposes limitations of current LLMs: they were not trained to treat specialist representations as native inputs, and access to information does not ensure its reliable interpretation, integration or use during reasoning. We call on specialist developers to provide consistent representation interfaces and, especially, on LLM developers to train models to receive, combine and reason with these inputs. Together, these efforts could expand the information available to scientific agents while preserving the flexibility of conventional tools. We release data, pretrained connectors, and code for training, evaluation and interactive inference: <https://github.com/PolymathicAI/AI-Mindmeld>.

1 Introduction

Large language models (LLMs) are powerful tools for interpreting scientific literature, developing hypotheses and reasoning about experiments. Tool calling expands their capabilities further, en-

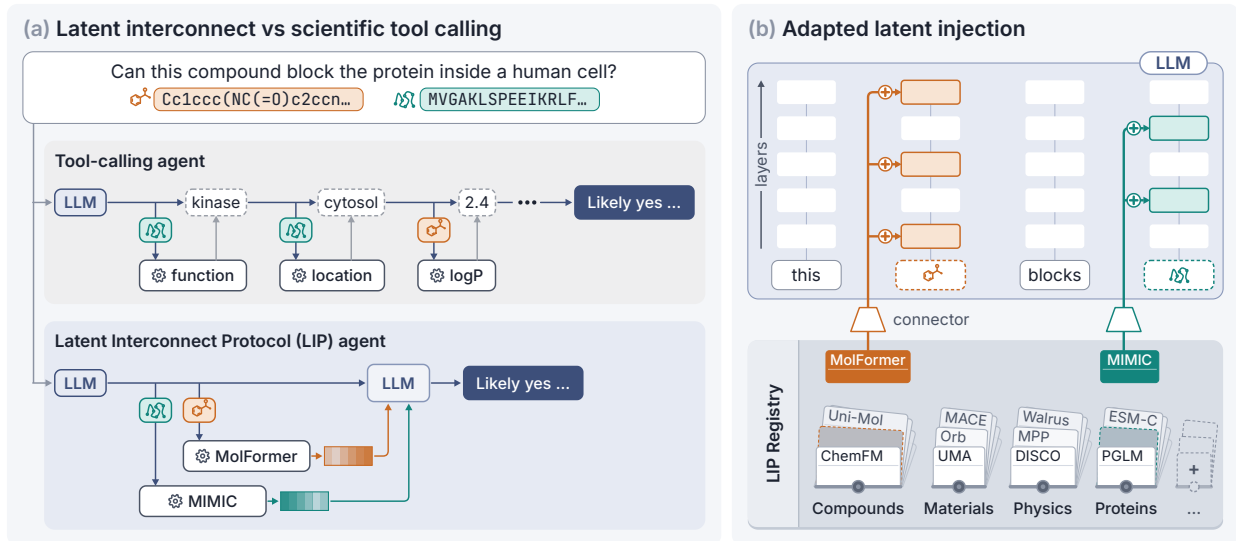


Figure 1: **Latent Interconnect Protocol alongside tool calling.** (a) The tool-output route returns selected properties as words or numbers; the latent route supplies specialist representations through learned connectors. The question and outputs are illustrative, and one agent can use both routes. (b) Connectors selected from the registry write the protein and compound representations into separate prompt slots at the marked layers. The $\oplus$ symbols denote the mixing operation defined in Section 2.1. The layer choices are schematic and may differ between specialists. Each representation occupies one token here; LIP also permits multiple-token slots. Registry entries are trained independently, and new connectors can be added without changing the shared language model or existing connectors. The displayed models and domains illustrate the registry’s organization.

abling them to act as autonomous agents that retrieve data, run calculations and interact with experimental systems [64, 78]. Such agents have been used to plan chemical syntheses [9], operate laboratory systems [8] and answer biomedical questions using external tools and databases [31]. They can also consult specialist machine-learning models trained on scientific data, gaining access to predictions and knowledge about proteins, molecules, cells and other systems [42].

Specialist models may produce many kinds of artifacts, but their contents can inform an LLM’s reasoning only if the LLM can ingest them into its context. For current-generation LLMs, this generally means text, images, video or audio—the communication media used by humans. Routing information between models through these formats imposes a severe bandwidth bottleneck, even though both models operate on high-dimensional representations (Figure 1a). As an example, for a complex query regarding a protein sequence, an agent may need to make explicit queries to specialized tools to calculate or retrieve properties. Making these properties explicit requires selecting them, retrieving or computing them, and adding the results to the agent’s context. Direct representation exchange offers a richer channel: a *mindmeld* in which an LLM receives specialist information in latent form. One representation can carry many properties together, including information the LLM does not know or would not think to request.

A framework for this richer exchange should preserve the strengths that make tool use powerful. It should be *extensible*, making new specialist models easy to add; *flexible*, allowing the same representations to support different questions and tasks; and *compositional*, enabling information from independently added specialists to be used together without training on each new combination.

Furthermore, frontier language models embody substantial training investment and continue to evolve through new releases [37, 53]. Requiring each specialist to adapt and maintain its own LLM would make community-wide extension costly to sustain. A practical interface should therefore keep the shared LLM frozen and confine alignment to connectors that are economical to fit and refit as models evolve. Together, these properties would allow a scientific agent to draw on a growing set of models as its questions change.

Prior work connects scientific representations to LLMs [49, 70, 83], while the closest framework for independently composed connections focuses on perceptual modalities [55]. Updating the LLM during alignment compromises independent extensibility by changing the shared interface [70], and task adaptation through in-context demonstrations is distinct from composing independently connected specialists without examples of their combination [83]. More fundamentally, these interfaces constrain specialist information to the input embedding layer [49, 70, 83, 55], despite evidence that deeper layers capture higher-level linguistic information [67] and that multilayer injection improves performance [52] (see also Section 4.5). What remains missing is a general framework for specialist models that jointly supports flexibility, compositionality and extensibility while allowing representations to enter beyond the input embedding layer.

We introduce the Latent Interconnect Protocol (LIP) to implement this form of communication. Under LIP, each specialist is linked to the LLM by a *connector*: a learned mapping from the specialist’s representation to the LLM’s internal states at designated positions in a prompt (Figure 1b). Our ablations support injecting representations into deeper layers: a single write at a selected internal layer outperforms a single input write, while injection at six layers gives the highest average performance (Section 4.5). In our implementation, connectors are trained independently while the specialists and shared LLM remain frozen, so a new specialist can be added without changing the LLM or existing connectors. A short sentence can then bring together representations from several specialists. The LLM’s existing knowledge and reasoning provide the basis for using these independently supplied representations together. LIP complements interfaces such as the Model Context Protocol (MCP) [4]; Python execution, database queries and other tools remain indispensable.

Our demonstration connects three protein specialists and three small molecule specialists to three LLMs, Gemma 3 (4B and 12B parameters) and Kimi K2.6 (1T total, 32B active parameters) [53]. These connections support questions about unseen objects and transfer to new question types. We also test whether independently trained connectors can combine information about the same protein and answer questions about relationships between proteins and compounds. Task performance and compositionality generally improve with LLM capability, with particularly strong gains in zero shot task transfer. Together, these demonstrations show how independently connected specialists can extend an LLM’s capabilities across changing questions and combinations of information sources.

Our implementation illustrates a broader limitation of connecting specialists to current LLMs: the LLM was not trained to interpret their representations as native inputs. Alignment can make specialist information accessible without ensuring that the LLM recognizes what kind of input it has received, combines it reliably with other sources, or uses it consistently during generation. For example, the LLM sometimes describes an injected protein representation as a familiar word or protein name, and generating a rationale before answering reduces protein property prediction performance across all three LLMs.

Realizing the broader vision calls for a community effort on both sides of the interface. For LLM developers, the opportunity is to make specialist representations a native part of model training, explicitly teaching models to receive, combine and reason with them. Specialist-model developers can support this direction by exposing representations through consistent, documented interfaces. Shared conventions would make new models easier to connect without requiring them

to learn the same representations. We call on the community to develop these capabilities so that scientific agents can use the information specialist models already encode, at the bandwidth of their representations and with the flexibility of ordinary tools.

Related work

We relate LIP to six areas of prior work: text-based model interfaces, LLMs that read specialist representations, support for multiple domains, support for multiple specialists within a domain, composition without joint training, and communication through internal states.

Interfaces through text. Beyond the tool-using agents discussed in the introduction, Socratic Models and HuggingGPT connect many pretrained models through a language model and exchange information among them as text [79, 65]. A second family writes biological objects themselves as strings and trains the language model on them [56, 76, 73, 74, 38, 60, 5]; having been trained on such mixtures, several of these models already answer questions that relate a protein to a small molecule in one prompt [56, 76, 73]. SciCore-Mol and Kamai et al. replace the text interface with a learned representation, but they fine-tune the language model and each serves one domain [16, 34].

Reading a specialist representation. Once a representation reaches the language model, it can be read and described in words. MetaLM proposed language models as a general-purpose interface to other foundation models, training the LLM together with its connections [30]. Later systems describe images, proteins, molecules, cells and stars from their embeddings [43, 13, 49, 63, 34, 1, 48, 84, 47, 41, 12]. Most of them fine-tune the language model, which can degrade its performance on text-only tasks [80]; those that keep it frozen typically either insert new modules [3, 82] or supply the representation only at the input [69, 57, 17, 33], as in BLIP-2 and ChatNT [40, 18]. DeepStack and ImageBind-LLM inject visual features at multiple layers while also adapting the LLM [52, 28]. Patchscopes and activation addition write vectors into the residual stream of an unmodified model to interpret or steer it [23, 71]; we use such a write as a trained interface at several layers of an unmodified model.

Several types of object. A language model that reads one specialist can in principle read several. PaLM-E and ChatBridge train on combinations of inputs [19, 81]. OneLLM aligns eight modalities through shared components, revisits earlier modalities’ data when adding new ones, and fine-tunes the LLM [27]. Scientific systems that align several types separately, such as BioMedGPT-10B and BioVERSE, evaluate one type per prompt [49, 70], and the general method Vector-ICL connects encoders from many domains but uses one encoder per task [83].

Several models of one type. The question of breadth has a counterpart within a single type of object, where different encoders capture different aspects of the same protein or molecule. Existing systems fuse several such encoders into one jointly trained input [72, 44, 36, 77, 1, 35], or offer alternative encoders that are used one at a time, either with the language model fine-tuned for each [20] or through separate projectors into one frozen language model [83]. We place separately aligned encoders side by side in separate slots of one prompt.

Composition without joint training. Placing several separately aligned representations in one prompt is useful only if the language model can combine them. That separately aligned inputs can be combined without joint training has been reported for perceptual modalities in a frozen language

model by X-InstructBLIP [55]. ImageBind-LLM transfers an image-trained connection to other modalities using their prealigned ImageBind space [28]. Outside language models, ImageBind aligns modalities through a common hub modality [24], and BioBridge learns knowledge-graph-supervised transformations between biomedical models for retrieval [75]. We evaluate independently aligned scientific inputs by combining representations of the same protein and answering protein–compound relationship questions, distinguishing assembly from useful joint use (Section 4.3).

Communication through internal states. More broadly, several studies let models communicate through internal representations instead of text [15, 62, 46]. Language models have exchanged output embeddings [58], intermediate states in Communicating Activations [59], key–value caches in Cache-to-Cache [22] and hidden states [85]; CALM composes two frozen language models through cross-attention trained on examples of the combined skill [7]; the Vision Wormhole connects vision-language models through their visual input pathway [45]; and frozen encoders have been connected through their latent spaces [54, 50]. Two of these studies argue explicitly that decoding an internal state into text discards information [59, 22], a related motivation for latent communication. Our emphasis is the bandwidth cost of selecting and rendering scientific information as explicit outputs. We develop a direct channel for specialist models that have no language interface, with a language model as the medium through which they communicate.

2 Latent Interconnect Protocol

The mindmeld outlined above calls for *extensibility*: new specialists should be easy to add; *flexibility*: their representations should support changing questions and tasks; and *compositionality*: independently connected specialists should work together without training on each new combination. These desiderata motivate a framework in which each specialist has its own connection to the LLM, task instructions remain separate from the representation interface, and multiple connections can be assembled in one prompt. The Latent Interconnect Protocol (LIP) provides one realization of this idea. Here, connectors are learned independently against a shared frozen LLM, using affine maps to write specialist representations into its internal states.

A connector binds a particular specialist to this interface: it records the representation to extract, the maps into the language model’s states, and where and how those maps are applied. Figure 1b illustrates a prompt assembled from such connections. Tools can invoke the specialists; LIP determines how their representations enter the LLM. Connectors can differ in architecture, injection layers and training procedure. Here, we describe one approach to constructing them, followed by how LIP registers and combines independently constructed connectors. Section 3 describes the connections constructed for six specialists and three LLMs. Figure 2 summarizes the construction procedure used here.

2.1 Connecting a specialist

A representation enters through designated placeholder tokens, which we call its slot. In Figure 1b, the protein and compound each receive a slot; the same object could also occupy several slots supplied by different specialists. LIP allows a slot to span multiple tokens. Our main experiments use one token per representation, with multiple tokens examined in Section 4.5. The rest of the prompt supplies ordinary language, as in “Tell me about the protein <slot>.” The same connector can be reused with different questions in the surrounding text. The connector writes at the slot, and subsequent tokens read its state through the language model’s existing attention.

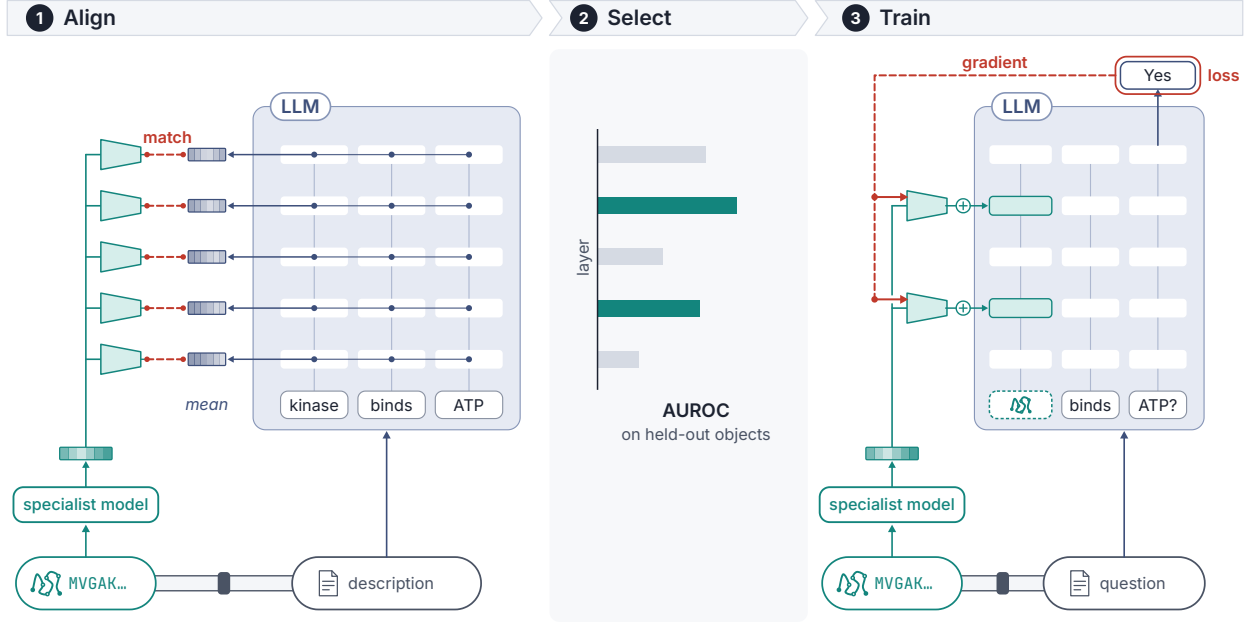


Figure 2: **Three stages of independent connector training.** **1. Align.** Layer-specific maps match specialist representations to text-derived reference states in the LLM. The schematic illustrates alignment to description states; the connectors evaluated here instead align to states obtained from object names. **2. Select.** Injection layers are selected to span the LLM’s depth, guided by local peaks in single-layer AUROC on held-out scan objects. **3. Train.** The selected maps write into the object’s slot and are refined by answer loss. Gradients pass through the frozen LLM to the connector; both pretrained models remain frozen throughout. Layer counts, bar lengths and example text are illustrative, not measured results or recorded model responses. Table 2 gives the actual schedules.

We implement the writes as follows. For a specialist embedding $e \in \mathbf{R}^{d_f}$, fixed coordinate-wise standardization gives $z = (e - \mu)/s$. At each selected layer ℓ , an affine map produces $\delta_\ell = W_\ell z + b_\ell \in \mathbf{R}^d$, where d is the language model’s residual width. These maps align the specialist’s representation with states usable by the LLM. Each connector specifies its representation, standardization and injection layers.

At each injection point marked by $\oplus$ in Figure 1b, the projected vector is mixed with the slot’s residual state h_ℓ at the input of layer ℓ :

$$h_\ell \leftarrow \|h_\ell\| \frac{u_\ell}{\|u_\ell\|}, \quad u_\ell = (1 - \alpha_\ell) \frac{h_\ell}{\|h_\ell\|} + \alpha_\ell \frac{\delta_\ell}{\|\delta_\ell\|}. \quad (1)$$

The mixing weight $\alpha_\ell \in [0, 1]$ interpolates between the two directions while preserving the incoming norm; the mixing weights at all selected layers are learned during the training stage. Only the slot is modified directly; its contents affect subsequent computation through the model’s normal processing. Our implementation supplies a learned, object-independent constant at the token embedding. Numerical implementation details are given in Supplementary Section A.4.2. On prompts without injected slots, the connector is inactive and the language model’s computation is unchanged.

2.2 Constructing an independent connector

We describe one approach to constructing a connector through alignment, injection layer selection and task training (Figure 2). Both pretrained models remain frozen throughout. Section 3 gives the specific choices used in our experiments.

We start from a dataset of objects paired with textual descriptions and a set of tasks with a metric for evaluating the LLM’s responses. For example, Yes/No questions can be evaluated by AUROC using scores derived from the LLM’s Yes and No token probabilities. The paired descriptions provide alignment targets, while the tasks provide feedback for layer selection and training.

1. Align. Alignment provides an initial mapping from specialist representations to states the LLM can use (Figure 2, stage 1). Our approach fits these maps to reference states produced when the LLM processes the paired descriptions. Because different LLM layers carry different information, each candidate injection layer receives its own map. This provides a starting point for subsequent task training.

2. Select. Writing at every layer can require billions of connector parameters. A practical connector therefore needs a limited set of injection layers that makes effective use of the specialist representation. Evaluating the aligned maps individually on these tasks using the chosen metric provides a performance profile across the LLM’s depth (Figure 2, stage 2). This profile can guide selection toward local performance peaks while retaining coverage of different depths, allowing the representation to enter at several stages of computation.

3. Train. Each connector is trained independently on these tasks to improve the LLM’s use of the specialist’s representations (Figure 2, stage 3). Gradients from the task loss pass through the frozen LLM to update the connector’s parameters while both pretrained models remain unchanged. Because trainable parameters and optimizer state are confined to the connector, adding a specialist does not require a separately adapted copy of the LLM: the same frozen LLM serves every connector registered for that version.

2.3 Registry and composition

The registry in Figure 1b collects connectors for the shared language model. A registry entry identifies a specialist model and LLM pair, the representation and standardization, and the learned maps and injection schedule. It contains what is needed to apply the connector without its training data. An entry can be added or updated by fitting only its connector, leaving the language model and other connectors unchanged. Application details are described in Supplementary Section A.6.

At inference, each slot selects its connector independently. Several slots can describe the same object through different specialists, or different objects from the same or different domains. The connectors write to their assigned positions, and the language model’s existing knowledge, attention and reasoning provide the means to combine their information. Sharing the LLM and slot interface allows these combinations to be assembled without further fitting. Their usefulness depends on whether the language model can use them together, which motivates the experiments that follow.

3 Constructing the demonstration

We demonstrate LIP by connecting protein and small molecule specialists to Gemma 3 (4B and 12B parameters) and Kimi K2.6 (1T total, 32B active parameters) [53]. These domains provide a concrete example of the broader opportunity for latent communication: specialist representations

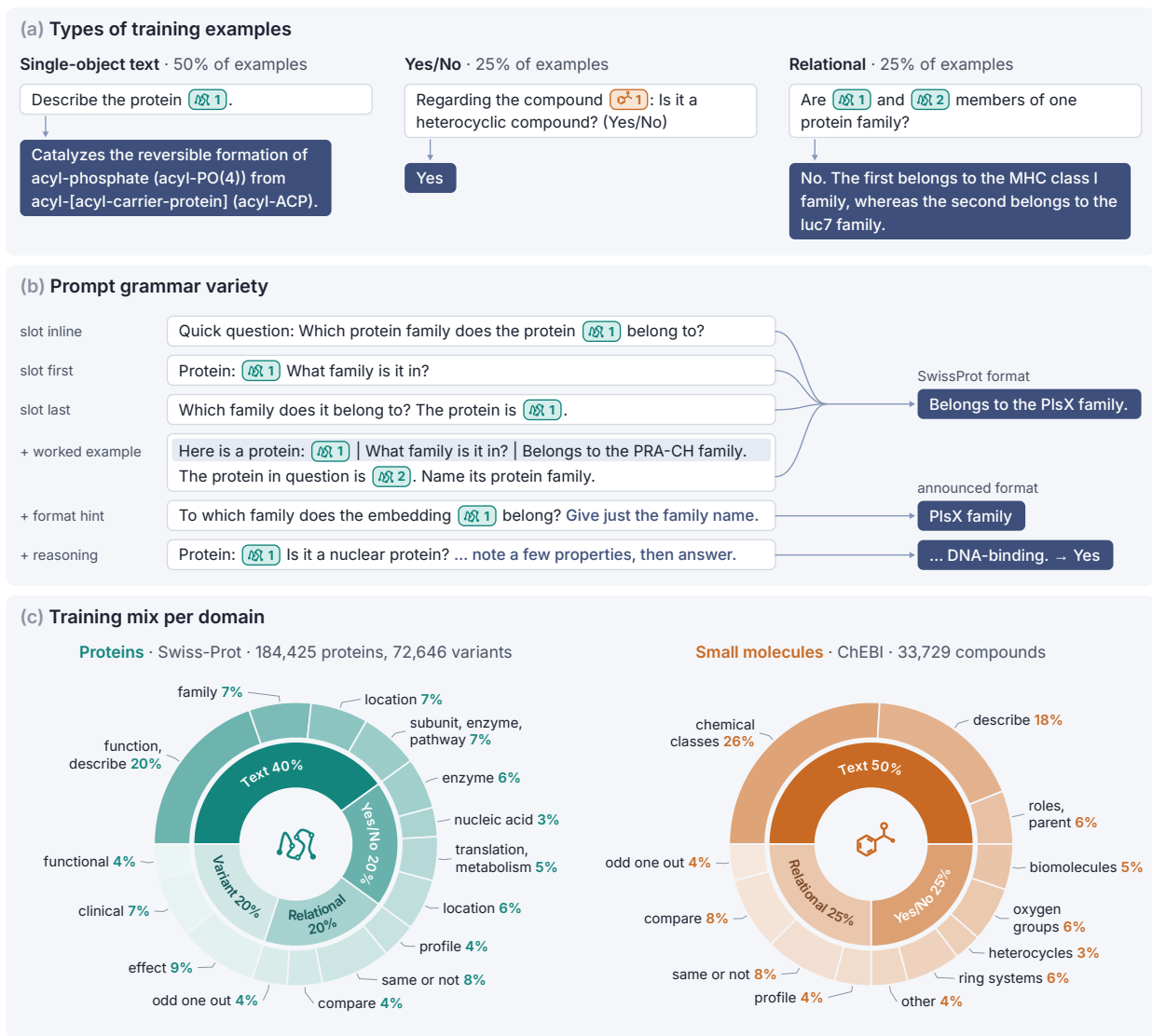


Figure 3: **Varied supervision through one connector.** (a) Configured shares among non-variant examples. Illustrative prompt–target combinations, not logged draws or generated answers; the description answer is abbreviated and the relational pair is schematic. The full description field is truncated at 48 tokens during training. (b) Prompt variations in slot position, worked context, answer format and reasoning. All numbered slots within a prompt use the same connector. (c) Inner rings show the mix of training streams; outer rings show the question types. Protein training additionally includes questions about mutation effects, pathogenicity and loss of function. Percentages refer to the full training mix in each domain. Headers give training-pool sizes; sampling details are in Supplementary Section A.2.3.

contain information relevant to many questions, and relationships between their objects create meaningful opportunities to combine that information. Questions about individual proteins or compounds test the reuse of each connection, while protein-compound questions test whether independently connected sources can work together.

For each LLM, we connect the same three specialists from each domain, fitting each connector independently while keeping all pretrained models frozen. This lets us examine reuse across independently learned representations and how it varies with LLM capability.

We evaluate extensibility through independently added specialists, flexibility through familiar and new questions about unseen objects, and compositionality through joint questions using independently trained connectors within and across domains. These evaluations also reveal how the ability to interpret and combine latent inputs varies with LLM capability, identifying what must improve to realize the broader vision.

3.1 Models and representations

The protein specialists are MIMIC [25], ESM-C [11] and PGLM [14], each receiving an amino-acid sequence. The molecule specialists are MoLFormer [61], Uni-Mol2 [32] and ChemFM [10].

For each specialist, we inspect read layer performance using property probes and, for proteins, variant probes. Based on these, we select up to two layers with complementary performance, retaining their concatenation when it improves performance on at least one task. These choices precede alignment, and the same selected representations are used across all three LLMs. Table 1 lists the representations and their dimensions. Read probes and layer profiles are given in Supplementary Section A.4.1.

Specialist	Representation	Dim.
Proteins		
MIMIC	Layer 16 (mean pooled) Layer 20 (mean pooled)	3,072
ESM-C 600M	Layer 33 (mean pooled) Layer 36 (mean pooled)	2,304
PGLM-1B	Layer 24 (mean pooled)	2,048
Molecules		
MoLFormer-c3	Layer 9 (mean pooled) Layer 12 (EOS token)	1,536
Uni-Mol2 1.1B	Layer 23 (CLS token) Layer 23 (mean pooled)	3,072
ChemFM-1B	Layer 17 (last token) Layer 17 (mean pooled)	4,096

Dim.: extracted representation width; ||: concatenation in the listed order.

Table 1: **Specialist read layers and representations.** Read layers were chosen using property probes and, for proteins, variant probes; complementary layers were concatenated when they improved performance on at least one task.

Each representation occupies one token in the main experiments. Full model and extraction details are given in Supplementary Section A.1.

3.2 Prompts, data and evaluation

Prompt construction. We use varied supervision so that each connector is trained to make an object’s representation useful for different questions. Single-object text targets ask for descriptive information, Yes/No targets focus on individual properties, and relational targets ask about multiple objects (Figure 3a). Reference answers are constructed from the corresponding annotations.

Both domains use these three example types, drawing on protein function, family and location annotations or compound definitions, classes and relationships. Protein training additionally includes questions about mutation effects, pathogenicity and loss of function; Figure 3c shows the training mix for each domain. All slots in relational prompts use the same connector, preserving independent connector training; combinations of different connectors are reserved for evaluation.

To encourage reuse across ways of asking a question, a compositional prompt grammar augments text and Yes/No examples by varying wording, object references, slot placement, requested answer format and reasoning. These variations are intended to make the representation usable across linguistic contexts while allowing the surrounding instruction to determine the response. Figure 3b illustrates variations in slot position, worked context, answer format and requests for reasoning. Worked examples use other training objects, with every slot supplied through the same connector. Supervision categories and sampling rules are in Supplementary Section A.2, with question examples in Section A.3.

Data split. To evaluate generalization beyond the objects used to fit a connector, we separate training and held-out pools before constructing examples (Figure 4a). Protein sequences and annotations come from Swiss-Prot, the reviewed part of UniProt [68]; compounds come from the three-star entries of ChEBI [51]. The molecule training pool is substantially smaller, with approximately 34,000 compounds compared with 184,000 proteins, and covers a narrower range of annotation types; protein supervision additionally includes variant effects (Figure 3). The protein split uses an MMseqs2 exclusion criterion of 30% sequence identity and 80% coverage [66]; molecules are separated at ECFP4 fingerprint Tanimoto similarity 0.5 after grouping related chemical forms. The exclusion applies to every object in a connector-training prompt, including worked examples and relational partners, so those contexts cannot reintroduce held-out objects.

Layer selection uses scan objects from the training pool, excluded from fitting the maps used in the scans. Split construction is detailed in Supplementary Section A.2.1. These exclusions concern connector training, not exposure during the LLM or specialist model’s pretraining.

Alignment and injection layer selection. We implement the *Align* and *Select* stages outlined in Section 2.2 as follows. We fit ridge maps from the selected specialist representations to LLM states at object names in prompts containing a description followed by the name. We average the name token states for proteins and use the final name token state for molecules. Names are needed for these fitting objects, but not when using the trained connector. Name and description reference targets are compared in Supplementary Section A.4.4.

We evaluate each aligned map separately on the scan objects using five diagnostic Yes/No questions per domain. Mean task AUROC gives a profile across injection layers for each connector; injecting the LLM’s own name token states provides a reference profile. For each domain and LLM, we take the maximum across these profiles at each layer to form an envelope. Guided by its local maxima, we select layers spanning the LLM’s depth and use the resulting schedule for all specialists in that domain. Table 2 lists the schedules, with profiles shown in Supplementary Figure 18.

At every selected injection layer, each connector’s mixing weight is initialized from its mean task AUROC above chance: we subtract 0.5, normalize by the best layer’s above-chance score and clip to [0.1, 0.9]. All these weights are then learned during training. Reference layer choices, diagnostic questions and fitting details are given in Supplementary Sections A.4.3 and A.4.5.

Training and evaluation tasks. We deliberately withhold selected task categories to test whether a trained connection can support new questions (Figure 4a). Task training uses the varied

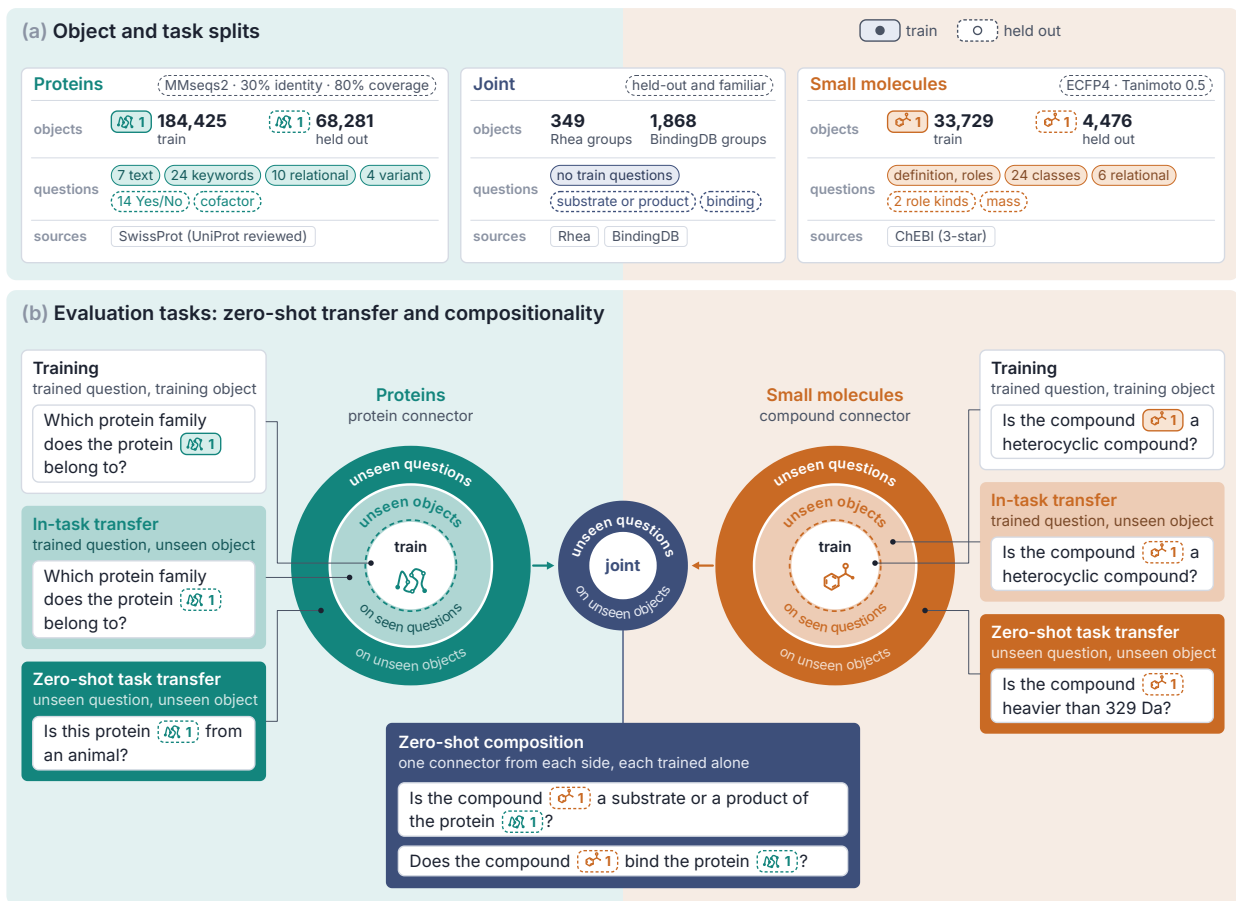


Figure 4: **Object, task and cross-domain evaluation design.** (a) Training and held-out object pools, question types and data sources. Training counts give the pools used by the fitted connectors; held-out counts give the full validation and test splits, not evaluated sample sizes. The joint card reports evaluation groups from Rhea and BindingDB for reaction participation and protein–compound binding. Cross-domain evaluations include both familiar and held-out objects. (b) The progression from trained questions about held-out objects to held-out questions about held-out objects, then cross-domain questions using independently trained connectors. Solid slots denote training objects and dashed slots denote held-out objects. The example questions are illustrative. Within-domain composition of different connectors is an additional evaluation described in the text. Final task definitions, scored populations and controls are in Supplementary Section B.

LLM	Proteins	Molecules
Gemma 3 4B	2, 7, 11, 17, 26, 33	4, 7, 12, 16, 21, 32
Gemma 3 12B	6, 11, 16, 23, 33, 44	4, 9, 17, 23, 31, 41
Kimi K2.6	7, 13, 22, 33, 49, 58	7, 13, 22, 32, 49, 57

Table 2: **LLM injection layers.** Shared layers for each domain and LLM were chosen to span its depth, guided by local peaks in the maximum AUROC envelope over connectors and the LLM’s own-state reference.

annotation-derived questions above, with no object-naming response target. Reserved tasks include protein lineage and length, selected compound roles and compound mass. This restricted supervision creates a test of zero-shot transfer; applications prioritizing performance on known tasks could train on a broader range of relevant questions, including categories withheld here. Selected grammar entries are also reserved to test new phrasings. Withholding a question does not remove every related clue from the remaining training annotations.

The evaluations distinguish three forms of reuse (Figure 4b). Trained questions on unseen objects test generalization beyond the fitting pool; new questions on unseen objects test flexibility beyond supervised tasks; and joint questions test composition of independently trained connections. Within-domain comparisons combine different protein connectors. Cross-domain comparisons use Rhea reaction-participation and BindingDB protein-compound binding questions, bringing protein and compound connectors together without any cross-domain training stream. These evaluations include both familiar and unseen objects, with the available combinations determined by task and filtering. The Rhea comparison in Figure 8c contains only unseen proteins: nearly every seen-protein pair was excluded because its relation was already stated in the protein connector’s training descriptions. Different connectors are combined only at evaluation, without further fitting. We separately assess how response mode affects use of the supplied information. Main comparisons use the final 2,000-step checkpoint of each connector.

3.3 Training implementation and cost

We train each connector with a target batch size of 85 examples per optimizer step. Answer-token losses are weighted by answer type, semantic segment and position within each segment, emphasizing informative content over predictable connecting text, and normalized by the total token weight. We use AdamW with learning rates for the maps, biases and input constant scaled to their initial magnitudes, a 20-step linear warmup, and a constant rate thereafter. Following the Trellis optimization recipe [21], we distribute expert computation across eight GPUs and pack examples to avoid processing padding, leading to $5.1\times$ and $1.19\times$ improvements in training-step throughput, respectively.

Alignment and layer selection took tens of minutes for Gemma and a few hours for Kimi K2.6, using precomputed specialist representations. Training for 2,000 updates took roughly three hours on one H100 for Gemma 3 4B, four hours on one H200 for Gemma 3 12B, and 12 hours on eight H200s for Kimi K2.6 per connector. Detailed training settings are given in Supplementary Section A.5.

4 Results

For each of the three LLMs, we train six connectors independently on a subset of objects and tasks, without exposure to other connectors. We evaluate generalization to unseen objects, zero-shot transfer to new tasks, and composition within and across domains, examining how these capabilities vary with LLM capability. We then examine the LLM’s responses to latent inputs, the effect of reasoning, and how injection depth and token count affect performance.

4.1 Within-task generalization

We first evaluate whether connectors can answer familiar types of questions on previously unseen objects. As described in Section 3.2, the proteins and molecules are excluded from connector fitting, and the questions use wording withheld from training. For each Yes/No question, we rank objects

by the LLM’s relative probability of Yes versus No and report the area under the receiver operating characteristic curve (AUROC), with 0.5 indicating chance discrimination. Scoring and uncertainty are described in Supplementary Section B.1.

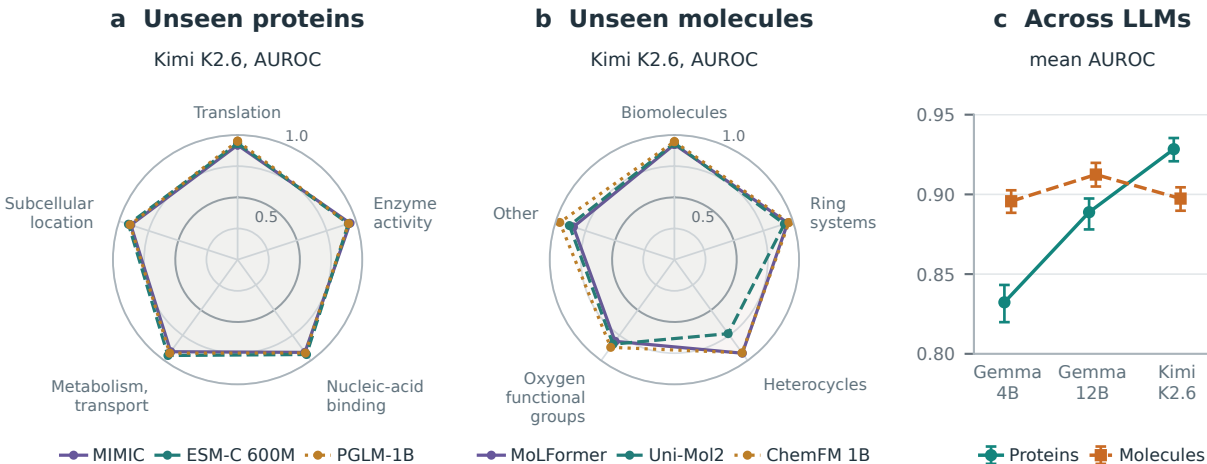


Figure 5: **Within-task generalization to unseen objects.** (a–b) Task-group AUROC for protein and molecule connectors attached to Kimi K2.6. Protein connectors perform similarly across groups, while molecule connectors differ most on heterocycles. (c) Mean AUROC across the three specialists in each domain for Gemma 3 4B, Gemma 3 12B and Kimi K2.6. Protein performance increases across these LLMs, whereas molecule performance varies less and peaks with Gemma 3 12B. Error bars in (c) show 95% bootstrap confidence intervals.

With Kimi K2.6, all six connectors support strong performance on objects excluded from connector fitting. Across the three protein connectors, mean AUROC over trained tasks ranges from 0.92 to 0.93, with similar performance across connectors on questions about enzyme activity, nucleic acid binding, translation, metabolism and transport, and subcellular location (Figure 5a). The three molecule connectors achieve mean AUROCs of 0.87–0.93 across trained tasks, with greater variation across chemical classes and the largest differences on heterocycles (Figure 5b). A matched control for unseen question wording is given in Supplementary Section B.2.

Averaging across the three specialists in each domain reveals different trends across LLMs (Figure 5c). Protein performance increases from Gemma 3 4B through Gemma 3 12B to Kimi K2.6. Molecule performance varies less and peaks with Gemma 3 12B. On new tasks, however, Kimi K2.6 substantially outperforms both Gemma models in both domains (Section 4.2).

4.2 Zero-shot transfer to new tasks

We next evaluate new tasks on unseen objects. Beyond explicitly excluding these question types from connector task training, we filter the free-text descriptions used as training targets to reduce disclosure of information related to these questions. We then classify a task as zero-shot only when the estimated remaining description exposure is below 0.1% of training examples (Supplementary Section B.3.1).

With Kimi K2.6, the three protein connectors show strong transfer across questions about lineage, magnesium binding, coiled-coil regions, acetylation and length (Figure 6a), with the highest scores for lineage or length.

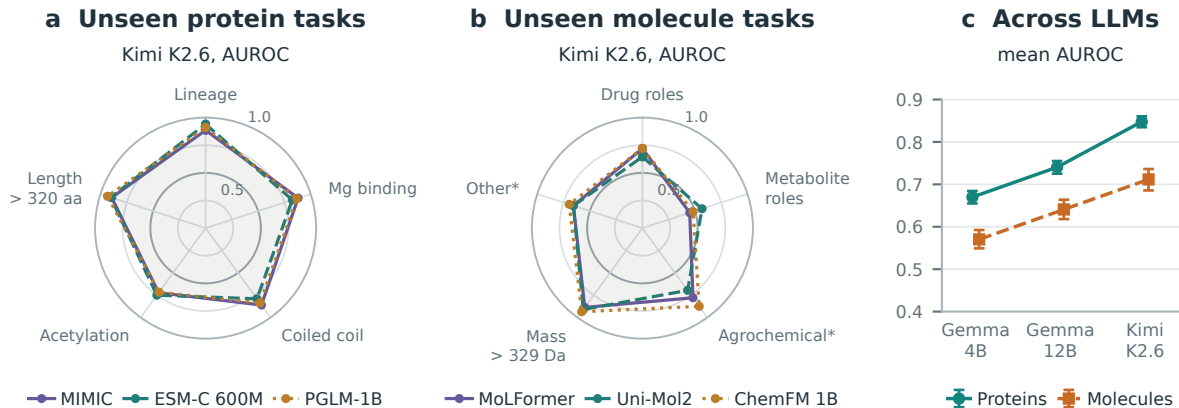


Figure 6: **Zero-shot transfer to new tasks on unseen objects.** (a–b) Task-group AUROC for protein and molecule connectors attached to Kimi K2.6. Protein connectors show similar transfer performance, while molecule transfer is strongest for mass. (c) Mean AUROC across the three specialists in each domain for Gemma 3 4B, Gemma 3 12B and Kimi K2.6. Performance increases across these LLMs in both domains. Error bars in (c) show 95% bootstrap confidence intervals. *Starred question types had exposure up to 0.3%, and in some cases related information may have been implied in the training examples’ free-text descriptions.

Molecule transfer is strongest for predicting whether mass exceeds 329 Da, with AUROC 0.89–0.93 across the three connectors (Figure 6b). Transfer to chemical roles is weaker and more variable: drug-role AUROC ranges from 0.65 to 0.72, while metabolite-role performance is near chance, at 0.45–0.57. The figure also reports agrochemical and other roles, including the starred question types with greater estimated description exposure. Task definitions and populations are given in Supplementary Section B.3.2.

Averaging across the three specialists in each domain, transfer performance increases from Gemma 3 4B through Gemma 3 12B to Kimi K2.6 (Figure 6c). The gains are evident in both domains, including molecules, for which performance on trained tasks varies much less across LLMs. These results show that the connections support questions beyond their task supervision, with stronger transfer in the more capable LLMs.

To examine what task training adds beyond alignment, Figure 7 compares the performance of the same connectors before and after *Train* (Section 2.2). Each point averages the three specialists in a domain for one LLM. Before task training, the aligned representations already support predictions on a range of trained and unseen questions.

For proteins, training improves performance across the displayed trained and unseen task groups with all three LLMs (Figure 7a–b). Molecule training generally improves performance on trained tasks, but reduces performance on most unseen tasks (Figure 7c–d). Mass prediction improves only with Kimi K2.6, illustrating that the effect of training also depends on the LLM.

These patterns suggest that connector training can improve general access to specialist information or favor the supervised properties at the expense of other information. The broad protein gains are consistent with improved access, while the molecule transfer losses suggest greater specialization toward the training tasks. Transfer may arise because the frozen LLM can combine accessible object properties with relationships learned during pretraining to answer new questions. A more capable LLM may be better able to use this information, consistent with the transfer trend in Figure 6c.

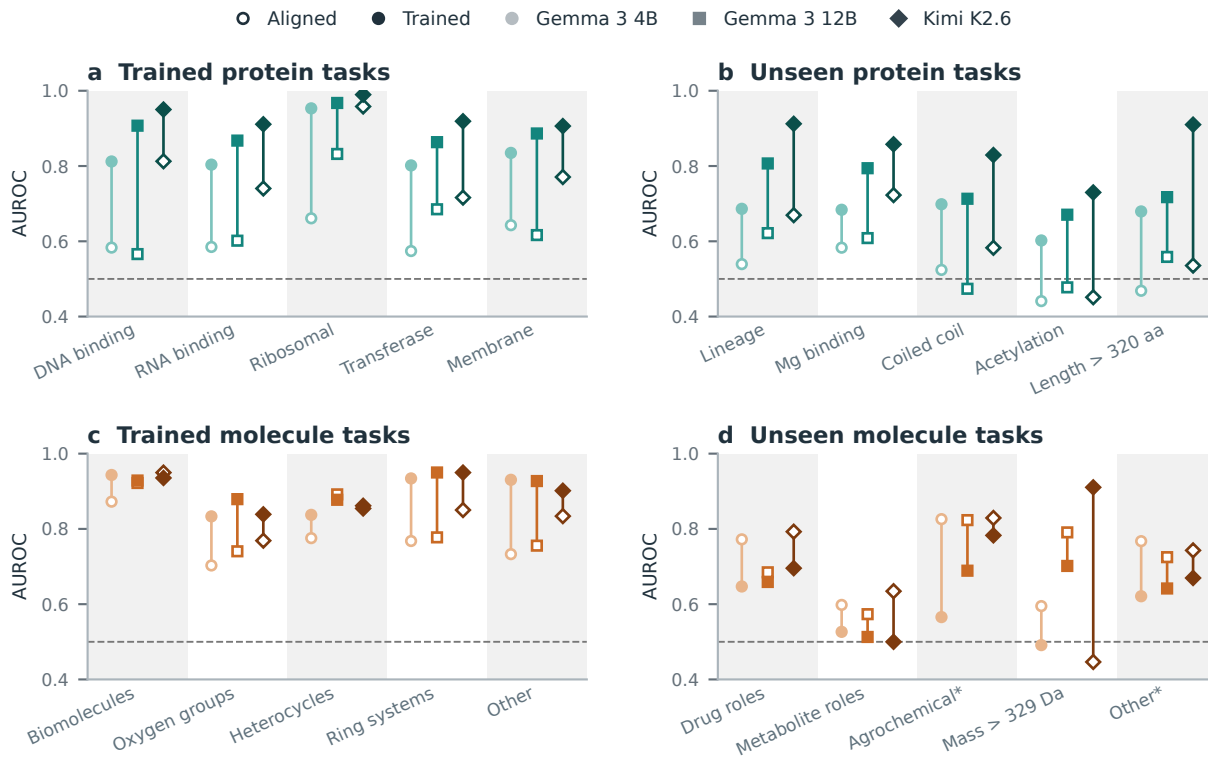


Figure 7: **Effects of connector training on trained and unseen tasks.** Open and filled markers show AUROC after alignment and after connector training, respectively, averaged across the three specialists in each domain for each LLM. **(a–b)** Training improves performance on both trained and unseen protein tasks. **(c)** Training generally improves performance on trained molecule tasks. **(d)** Performance declines on most unseen molecule tasks; mass prediction improves only for Kimi K2.6. *Starred question types had exposure up to 0.32%, and in some cases related information may have been implied in the training examples’ free-text descriptions.

We next ask whether independently learned connections can also be used together.

4.3 Zero-shot compositionality

Compositionality requires independently trained connections to remain useful when brought together. We test two forms of this capability: combining different specialist views of the same object, and answering questions about relationships between objects from different domains. Neither combination receives joint training.

For information merging, we supply three representations of the same unseen protein, one each from MIMIC, ESM-C and PGLM. We compare each connector individually with all three used together, asking trained and unseen questions with Gemma 3 4B, Gemma 3 12B and Kimi K2.6. Controls using repeated copies of one specialist are reported in Supplementary Section B.5.

The benefit of combining representations depends on the LLM (Figure 8a–b). With Gemma 3 4B, the combination performs worse than every individual connector on both trained and unseen tasks. With Gemma 3 12B, it approaches the best individual connector on trained tasks but remains below all three on unseen tasks. With Kimi K2.6, combining the representations improves mean AUROC over every individual connector on both task groups. Thus, the most capable LLM

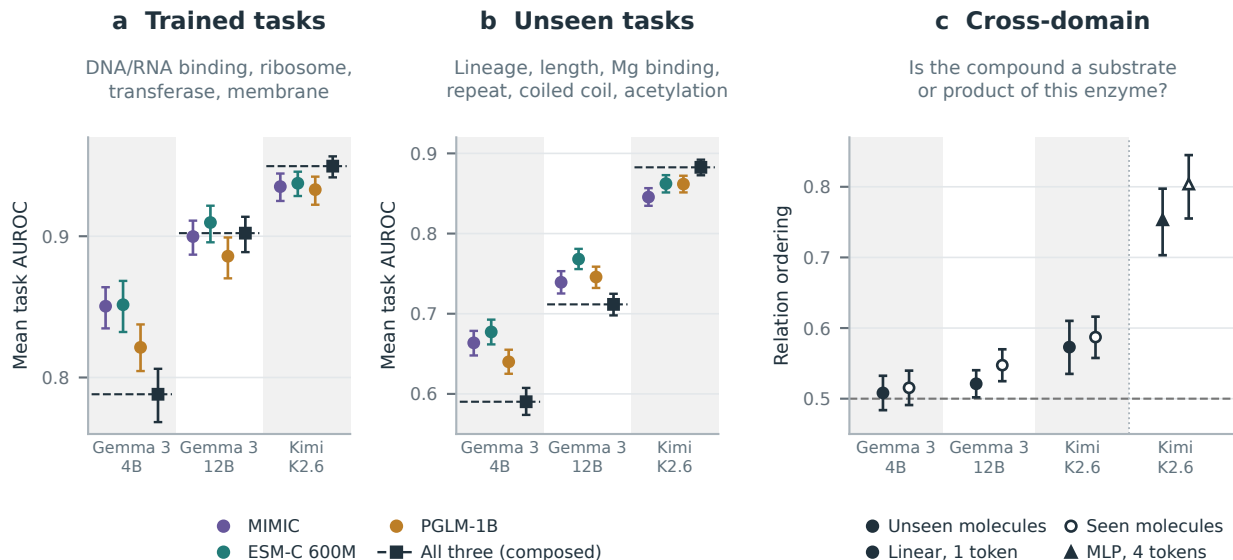


Figure 8: **Zero-shot composition of independently trained connectors.** Compositionality improves with LLM capability, with Kimi K2.6 benefiting most from combining independently trained connectors. **(a–b)** Individual and combined protein connectors on trained and unseen tasks. **(c)** Rhea reaction-participation ordering. The first three columns compare LLMs using ESM-C 600M and ChemFM 1B with single-token linear connectors. The final column shows Kimi K2.6 using ESM-C 6B and ChemFM 3B with four-token MLP connectors (Section 4.5). Point estimates are higher for seen molecules. All proteins are unseen: nearly every seen-protein pair was excluded because the protein connector’s training descriptions already stated the queried relation. Error bars show 95% bootstrap confidence intervals.

tested can benefit from multiple specialist representations without joint connector training, while additional inputs can reduce performance in smaller LLMs.

We next test composition across domains by connecting ESM-C 600M and ChemFM 1B to each LLM. Given a protein embedding and a compound embedding, can the LLM tell whether the protein catalyzes a reaction in which the compound is a substrate or product? We construct this task from UniProt-linked annotations in Rhea, a curated database of biochemical reactions [6]. Each matched group starts with two annotated protein–compound pairs; swapping their compounds gives two mismatched pairs made from the same objects. We measure how often the LLM scores an annotated pair above a mismatched pair sharing its protein or compound. Because each object appears in both types of pair, this tests their relationship rather than a preference for particular objects; chance ordering is 0.5. Task populations and scoring are detailed in Supplementary Section B.6.

All evaluated proteins are unseen. We exclude relationships stated in the protein connector’s training descriptions, which leaves almost no eligible pairs containing familiar proteins. We therefore compare unseen proteins paired with either familiar or unseen molecules (Figure 8c). Both settings test a new relationship question: the connectors were trained independently, without protein–compound relationship questions or access to one another’s representations.

With the single token linear connectors, ordering performance increases from Gemma 3 4B through Gemma 3 12B to Kimi K2.6. Gemma 3 4B performs near chance, whereas Kimi reaches approximately 0.57 with unseen molecules and 0.59 with familiar molecules. Using the larger ESM-C 6B and ChemFM 3B specialists with four token MLP connectors (Section 4.5), Kimi reaches 0.75 and 0.80, respectively, showing the potential of LIP beyond single token linear injections.

Familiar molecules have higher point estimates in every configuration. Together with the protein composition results, this shows that independently trained connectors can support joint use within and across domains.

4.4 LLM responses to latent inputs and the effect of reasoning

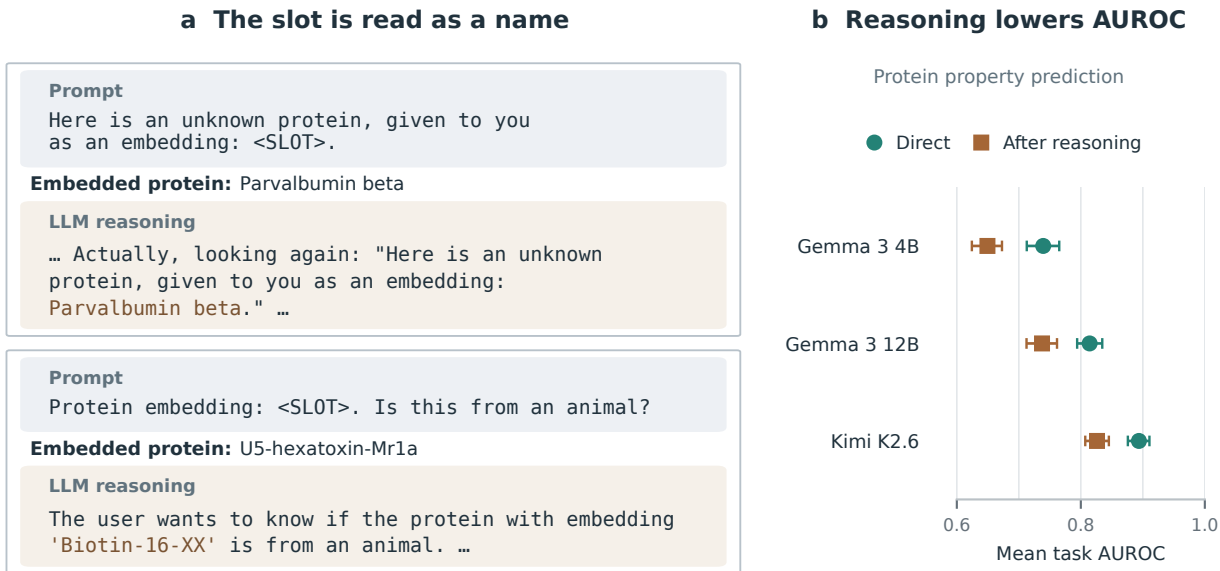


Figure 9: **LLM responses to latent inputs and the effect of reasoning.** (a) Examples of the LLM treating injected protein embeddings as protein names. (b) Mean task AUROC for direct answers and answers after reasoning, averaged across protein connectors for each LLM. Reasoning reduces protein-property prediction performance for all three LLMs. Bars in (b) show 95% object bootstrap confidence intervals, using shared draws across tasks and connectors.

The preceding results show that the LLM can use injected embeddings to predict properties and relationships. However, this form of input is out of distribution for the pretrained LLM and can also lead to undesirable behavior. During reasoning with a protein connector, for example, the LLM regularly treats the injected slot as though the user had supplied a protein name (Figure 9a). This interpretation is unsurprising because the LLM did not encounter these protein embeddings during its own training. Mapping an embedding to a familiar protein name is itself a nontrivial feat: it connects a specialist model’s representation to the extensive biological knowledge available to frontier LLMs. This raises a central question: does LIP work by mapping an embedding to a protein name and recalling facts associated with it, or does the embedding provide useful information beyond that name?

We first examine reasoning, during which this interpretation of the embedding as a name becomes explicit. Averaged across protein connectors, reasoning reduces mean task AUROC for Gemma 3 4B, Gemma 3 12B and Kimi K2.6 (Figure 9b). We request brief reasoning with a 200-token generation limit and then score Yes/No answers, retaining cases that reach the limit. The comparison therefore measures discrimination under this reasoning procedure, rather than the accuracy of naturally completed answers. In a separate control, when Kimi K2.6 receives the protein’s true name without latent injection, reasoning increases mean task AUROC by 0.03.

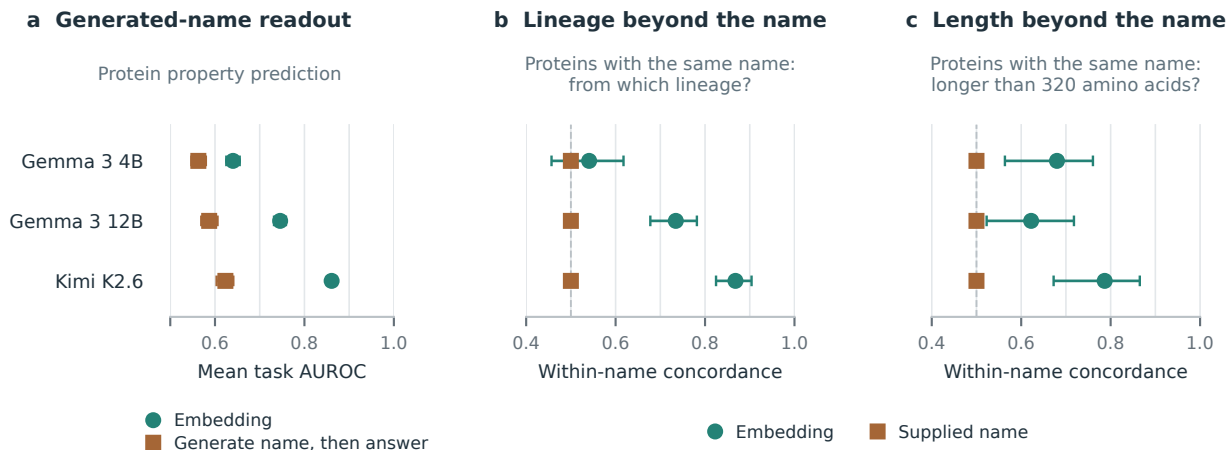


Figure 10: **Latent inputs preserve useful distinctions beyond names.** (a) Protein-property prediction using embeddings or names generated from those embeddings. Retaining the embedding gives higher AUROC for all three LLMs. (b–c) Lineage and length discrimination among proteins sharing the same name, measured by within-name concordance; lineage uses four low-exposure questions, and length tests whether a protein exceeds 320 amino acids. Embeddings provide information beyond the shared name, with Kimi K2.6 showing the strongest performance in both tests. Supplied names score 0.5 by construction. Error bars show 95% bootstrap confidence intervals.

To test whether this name readout accounts for LIP’s performance, we ask the LLM to infer a name from the protein embedding, then replace the embedding with that name and ask the same downstream questions. This replacement reduces mean task AUROC for all three LLMs (Figure 10a). This provides evidence that the embedding provides useful information beyond what the LLM recalls from the inferred name. However, the loss could reflect incorrect name inference rather than information missing from a correct name.

We therefore next ask whether the embedding provides useful information beyond even the correct supplied name. Among proteins with identical database-name strings, we test four low-exposure lineage questions and whether length exceeds 320 amino acids (Figure 10b–c). A deterministic name-only score must tie within each group, since both the name and the question are identical. Embeddings yield mean within-name concordance above 0.5 for both tests with all three LLMs, with Kimi K2.6 performing best in each. These results show that the latent input supplies useful distinctions beyond the shared name. Name-substitution and within-name controls are defined in Supplementary Section B.7.

Together, these results show that LIP’s usefulness cannot be explained solely by recalling information associated with the supplied object name. One interpretation of the reasoning decline is that the LLM maps the unfamiliar latent input to a familiar name and then reasons mainly from that name, leaving other useful information underused. Given the central role of reasoning in modern LLM use, this points to a need for frontier language models trained to work with information-rich latent placeholders and draw on their content throughout the reasoning process.

4.5 LIP ablations, extensions and scaling

Throughout the preceding experiments, each connector represented an object with one token, injected at multiple LLM layers. The specialists vary in size; when multiple versions were available, we selected the one closest to one billion parameters. We now examine how injection depth, token

count, specialist scale and connector architecture affect performance, and compare the resulting connectors with supervised linear baselines.

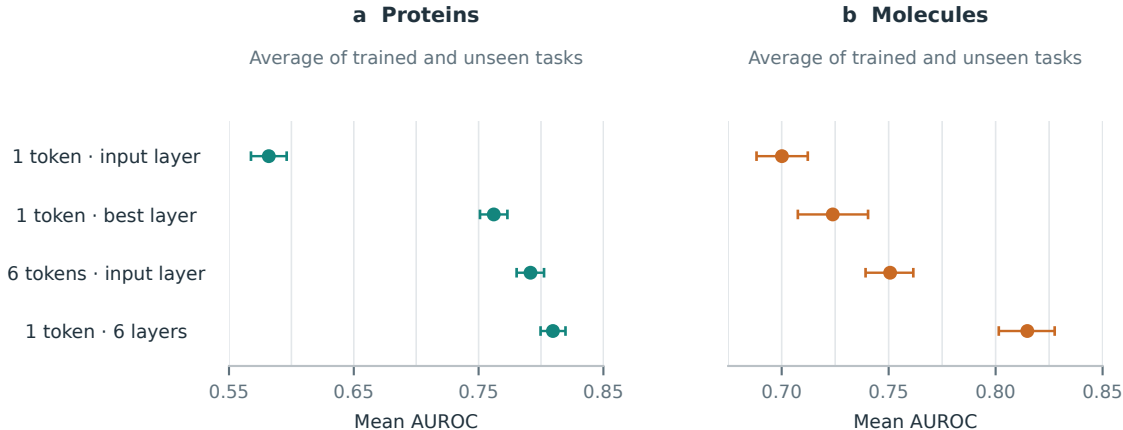


Figure 11: **Effects of write location and token count.** Gemma 3 12B with MIMIC (a) and ChemFM-1B (b). Mean of trained and unseen task performance; error bars show 95% bootstrap intervals. The paper’s connector uses one token and six injection layers. Moving a single-token write from the input to the best layer improves performance, supporting deep injection. Six input tokens match the six-layer connector’s parameter count but require more inference compute; even so, performance is higher than single-token injection at the best layer but below the six-layer connector.

Injection depth. We examine how injection depth and token count affect performance using Gemma 3 12B with MIMIC for proteins and ChemFM 1B for molecules (Figure 11). In both domains, moving a single-token write from the input to the best internal layer improves mean AUROC. Writing to the same token at six selected layers improves performance further, supporting the use of deep, repeated injection in the connectors used throughout the paper.

To compare repeated injection with a larger input interface, we also supply six tokens at the input layer. This matches the connector parameter count of one token with six injection layers, but requires more inference compute because the additional tokens pass through the LLM. Despite this additional compute, six input tokens achieve higher mean AUROC than one token at the best layer but remain below one token at six layers. These results suggest that, at matched connector parameter count, distributing injection across layers is more effective than supplying additional tokens at the input.

Beyond single token linear injections. LIP permits richer interfaces than a single token with linear injection maps. To explore these extensions with the aim of maximizing performance, we connect ESM-C 6B to Kimi K2.6 and compare four-token connectors with linear maps, MLPs and transformer blocks (Figure 12). We also compare using the same injection layers for all tokens with staggering their injection layers. Mean AUROC across trained and unseen tasks increases from approximately 0.89 for the single-token connector to 0.90 with four-token linear maps and 0.92 with MLP or transformer connectors.

The MLP performs comparably to both transformer variants. After 2,000 training steps, the transformer blocks’ attention maps remain fairly uniform, suggesting limited specialization in how

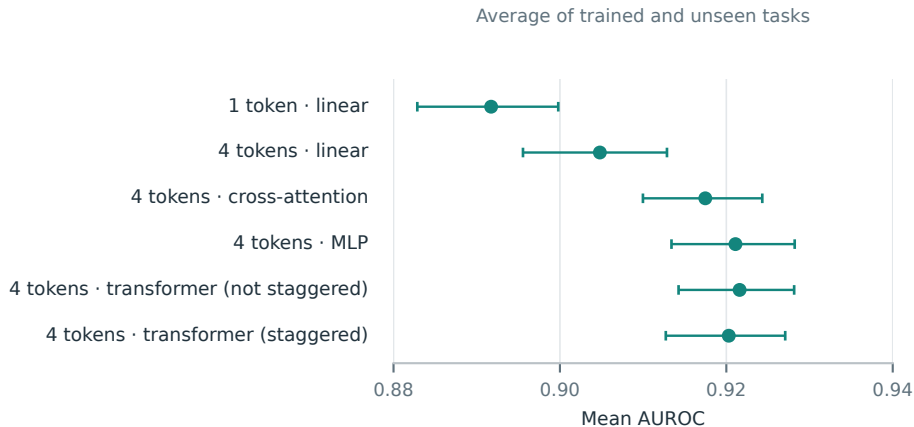


Figure 12: **Connector extensions.** ESM-C 6B with Kimi K2.6: mean AUROC across trained and unseen tasks after 2,000 training steps. Error bars show 95% bootstrap intervals.

they combine their inputs. This may help explain their similar performance to the MLP and suggests that realizing the benefits of more expressive connectors may require further training, changes to their optimization, or new residue-specific training tasks.

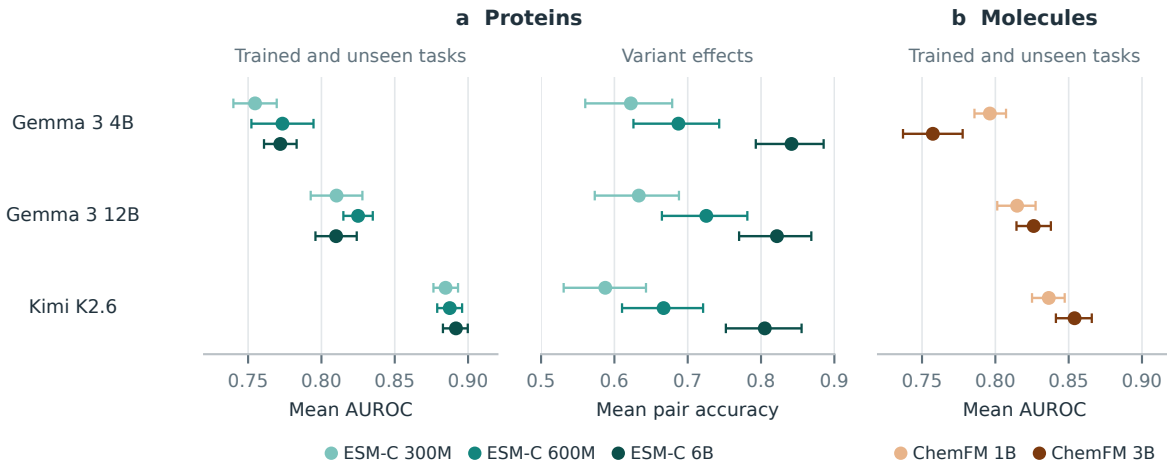


Figure 13: **Specialist and LLM scaling.** (a) Proteins: mean task AUROC and variant pair accuracy; (b) molecules: mean task AUROC. Protein property performance varies more across LLMs than across specialist sizes, while variant effects show the opposite pattern. Error bars show 95% confidence intervals, including seed variation for Gemma 3 4B and 12B.

Specialist and LLM scaling. We next examine how performance depends on the scale of both the specialist and the LLM, considering proteins (Figure 13a) and molecules (Figure 13b). For proteins, we compare ESM-C models from 300M to 6B parameters across the three LLMs. Protein property performance varies more across LLMs than across specialist sizes. This may partly reflect the larger difference in scale between the LLMs: Kimi K2.6 has 1T total and 32B active parameters, compared with 4B or 12B for Gemma 3. Architecture and pretraining also differ, so this comparison does not isolate parameter count.

To test how scaling affects sensitivity to small sequence changes, we also evaluate variant effects. We compare pairs of mutations from the same protein with opposite pathogenicity or loss of function labels. Pair accuracy measures how often the positively labeled variant receives the higher Yes/No log odds, with ties contributing one half.

This evaluation reveals a different scaling pattern. Increasing ESM-C from 600M to 6B yields substantial gains across all three LLMs, whereas increasing LLM size does not consistently improve performance. The relative benefit of specialist and LLM scaling therefore depends on the task: a more capable LLM can make better use of available specialist information, but the specialist must first encode the distinctions needed to answer the question.

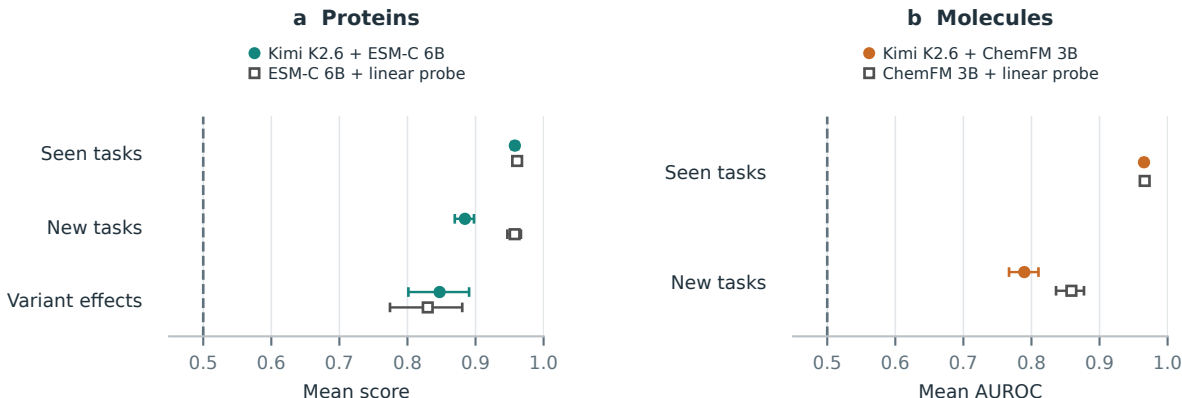


Figure 14: **Comparison with supervised linear probes.** Kimi K2.6 results using the four-token MLP connector of the previous section. On trained tasks, connectors match logistic regression in both domains. On new tasks, connectors trail the supervised probes, but are evaluated zero-shot. Scores are mean task AUROC or variant pair accuracy; error bars show 95% bootstrap intervals.

Comparison with linear baselines. The preceding results establish that LIP supports non-trivial prediction across diverse tasks. To put this performance in context, we compare the LLM connectors with supervised linear probes: logistic regression classifiers trained to predict each task’s labels directly from the same frozen specialist embeddings, without an LLM. On trained tasks, the LLM connector matches the supervised probe in both domains (Figure 14).

On new tasks, the LLM connectors underperform the probes in both domains, but this is not a comparison under matched supervision. Each probe is trained on labels for the task being evaluated, whereas the connector must answer these questions without explicit training on those tasks. The probes therefore provide a supervised reference for the information accessible in the specialist embeddings, while the connectors are being tested on zero-shot task transfer. For protein tasks, mean zero-shot AUROC is 0.88, compared with 0.96 for the supervised probe.

5 Discussion and outlook

LIP demonstrates a direct, high-bandwidth channel from scientific specialists to a shared language model: one latent input supports many questions, and independently aligned inputs can be used together. The broader position is that communication between models should not require every useful property to be selected and rendered for a human reader first. Keeping the LLM and its existing connections fixed makes extension possible through independent connector training, allowing a new specialist to join without changing those already connected.

The demonstrations establish three forms of reuse. Connections retain useful information about objects excluded from connector fitting; fixed connections support questions beyond their training tasks; and separately trained connectors support joint questions in combinations never fitted together, including protein–compound relationships. The quality of these predictions depends on both LLM capability and specialist scale. More capable LLMs generally provide better backbones for trained and zero-shot tasks, with the clearest gains in task transfer and composition. On trained tasks, the four token MLP connectors match supervised linear probes in both domains.

Limitations. Despite these gains, our experiments explore only a small part of the possible connector designs. More expressive transformer connectors do not yet outperform MLPs, and their attention remains fairly uniform after 2,000 training steps (Section 4.5). Further extensions must account for the extra inference cost of additional tokens and, for residue-level or spatial questions, the need to preserve finer detail in specialist representations.

Our scaling comparisons span LLMs with different model families and training histories, so they do not isolate parameter count as the cause of improved performance. Experiments spanning more specialists, scientific domains and LLMs will also be needed to establish the breadth of the framework’s usefulness. Evaluating these connections in scientific workflows would help determine how their benefits compare with the costs of constructing and maintaining them.

Our connectors also supply specialist representations to an LLM that was not trained to receive them. Connector training can improve performance on supervised tasks while reducing transfer, as observed for molecules. Together with the LLM’s uneven use of specialist information during reasoning and composition, this points to a remaining challenge: enabling independently connected specialists to contribute reliably to the LLM’s answers across changing questions and combinations.

Outlook. Preparing LLMs for representation exchange during their own training could enable more effective use of specialist information. Beyond learning to receive latent inputs, models could learn to reason in latent space, working with complex concepts that are difficult to express in human language. Emerging approaches to continuous and modality-grounded reasoning provide starting points for exploring this possibility [29, 39]. Such capabilities would extend the benefit of direct communication into the reasoning that follows it.

For LLM developers, concrete steps include pretrained slots for external representations, extending the reserved placeholders that already carry numerical values [26] or a multilingual encoder’s sentence embedding [2], and training objectives that reward their use across questions and combinations of sources. Preparing a shared LLM in this way could make it more receptive to specialists connected later through independent alignment, preserving extensibility without retraining the LLM for each addition. Documented, versioned injection interfaces would help sustain these connections as models evolve, making connectors easier to validate and refit across releases.

Specialist developers can contribute consistent extraction interfaces and clear descriptions of what their representations encode, including their granularity. These conventions would make models easier to connect while leaving them free to learn different representations. Connectors would supply the alignment, preserving the diversity of information available to the LLM.

Together, these efforts could support a growing registry of independently connected specialists alongside existing tool interfaces (Figure 1b). MCP, retrieval and Python would remain indispensable for accessing sources, running computations and acting on the world; LIP would enrich the information available within the agent’s context. We call on the community to develop models and interfaces that let scientific agents receive and reason with specialist knowledge at the bandwidth its representations support.

Acknowledgments

We would like to acknowledge the support of the Simons Foundation and of Schmidt Sciences. This work was supported in part by the AI2050 program at Schmidt Sciences (Grant G-25-70028). Additionally, computations were run at facilities supported by the Scientific Computing Core at the Flatiron Institute, a division of the Simons Foundation. We would like to thank François Lanusse, Irina Espejo Morales, Henri Palacci, and Roshan Rao for inspiring conversations pertaining to this project.

References

- [1] Hadi Abdine, Michail Chatzianastasis, Costas Bouyioukos, and Michalis Vazirgiannis. Prot2Text: Multimodal Protein’s Function Generation with GNNs and Transformers, 2023. AAAI 2024, 38(10):10757–10765, doi:10.1609/aaai.v38i10.28948.
- [2] Rajan Agarwal and Aarush Gupta. Languages are Modalities: Cross-Lingual Alignment via Encoder Injection, 2025.
- [3] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikołaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a Visual Language Model for Few-Shot Learning, 2022. NeurIPS 2022.
- [4] Anthropic. Introducing the Model Context Protocol. <https://www.anthropic.com/news/model-context-protocol>, November 2024. Announcement, 25 November 2024. Specification: <https://modelcontextprotocol.io/specification/2025-06-18>.
- [5] Lei Bai, Zhongrui Cai, Yuhang Cao, Maosong Cao, Weihang Cao, Chiyu Chen, Haojiong Chen, Kai Chen, Pengcheng Chen, Ying Chen, et al. Intern-S1: A Scientific Multimodal Foundation Model, 2025.
- [6] Parit Bansal, Anne Morgat, Kristian B. Axelsen, Venkatesh Muthukrishnan, Elisabeth Coudert, Lucila Aimò, Nevila Hyka-Nouspikel, Elisabeth Gasteiger, Arnaud Kerhornou, Teresa Batista Neto, Monica Pozzato, Marie-Claude Blatter, Alex Ignatchenko, Nicole Redaschi, and Alan Bridge. Rhea, the reaction knowledgebase in 2022. *Nucleic Acids Research*, 50(D1):D693–D700, 2022.
- [7] Rachit Bansal, Bidisha Samanta, Siddharth Dalmia, Nitish Gupta, Shikhar Vashishth, Sriram Ganapathy, Abhishek Bapna, Prateek Jain, and Partha Talukdar. LLM Augmented LLMs: Expanding Capabilities through Composition, 2024. ICLR 2024; CALM.
- [8] Daniil A. Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624:570–578, 2023.
- [9] Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D. White, and Philippe Schwaller. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 6:525–535, 2024.

- [10] Feiyang Cai, Katelin Zacour, Tianyu Zhu, Tzuen-Rong Tzeng, Yongping Duan, Ling Liu, Srikanth Pilla, Gang Li, and Feng Luo. ChemFM as a scaling law guided foundation model pre-trained on informative chemicals. *Communications Chemistry*, 9:3, 2025. Preprint arXiv:2410.21422.
- [11] Salvatore Candido, Thomas Hayes, Alexander Derry, Roshan Rao, Zeming Lin, et al. Language Modeling Materializes a World Model of Protein Biology. bioRxiv preprint, 2026. ESM C.
- [12] He Cao, Zijing Liu, Xingyu Lu, Yuan Yao, and Yu Li. InstructMol: Multi-Modal Integration for Building a Versatile and Reliable Molecular Assistant in Drug Discovery, 2023.
- [13] Eli M Carrami and Sahand Sharifzadeh. PQA: Zero-shot Protein Question Answering for Free-form Scientific Enquiry with Large Language Models, 2024.
- [14] Bo Chen, Xingyi Cheng, Pan Li, Yangli ao Geng, Jing Gong, Shen Li, Zhilei Bei, Xu Tan, Boyan Wang, Xin Zeng, Chiming Liu, Aohan Zeng, Yuxiao Dong, Jie Tang, and Le Song. xT-rimoPGLM: unified 100-billion-parameter pretrained transformer for deciphering the language of proteins. *Nature Methods*, 22(5):1028–1039, 2025.
- [15] Siyi Chen, Xiaoyan Zhang, Meng Wu, Jonathan Tremblay, Valts Blukis, Stan Birchfield, Rene Vidal, Alvaro Velasquez, Sijia Liu, and Qing Qu. See What I See, Know What I Think: Dense Latent Communication Across Heterogeneous Agents, 2026.
- [16] Yuxuan Chen, Changwei Lv, Yunduo Xiao, Zhongjing Du, Daquan Zhou, Yukun Yan, Zheni Zeng, and Zhiyuan Liu. SciCore-Mol: Augmenting Large Language Models with Pluggable Molecular Cognition Modules, 2026.
- [17] Xin Cheng, Xun Wang, Xingxing Zhang, Tao Ge, Si-Qing Chen, Furu Wei, Huishuai Zhang, and Dongyan Zhao. xRAG: Extreme Context Compression for Retrieval-augmented Generation with One Token, 2024. NeurIPS 2024.
- [18] Bernardo P. de Almeida, Guillaume Richard, Hugo Dalla-Torre, Christopher Blum, Lorenz Hexemer, Priyanka Pandey, Stefan Laurent, Chandana Rajesh, Marie Lopez, Alexandre Lat-erre, Maren Lang, Uğur Şahin, Karim Beguir, and Thomas Pierrot. A multimodal conversational agent for DNA, RNA and protein tasks. *Nature Machine Intelligence*, 7:928–941, 2025.
- [19] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. PaLM-E: An Embodied Multimodal Language Model, 2023.
- [20] Adibvafa Fallahpour, Andrew Magnuson, Purav Gupta, Shihao Ma, Jack Naimier, Arnav Shah, Haonan Duan, Omar Ibrahim, Hani Goodarzi, Chris J. Maddison, and Bo Wang. BioReason: Incentivizing Multimodal Biological Reasoning within a DNA-LLM Model, 2025.
- [21] Addie Foote, Rudolf Laine, and Tim Kostolansky. Post-training 50x faster. Workshop Labs, <https://www.workshoplabs.ai/blog/post-training-50x-faster>, March 2026. March 12, 2026.

- [22] Tianyu Fu, Zihan Min, Hanling Zhang, Jichao Yan, Guohao Dai, Wanli Ouyang, and Yu Wang. Cache-to-Cache: Direct Semantic Communication Between Large Language Models, 2025. ICLR 2026.
- [23] Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. Patchscopes: A Unifying Framework for Inspecting Hidden Representations of Language Models, 2024. ICML 2024.
- [24] Rohit Girdhar, Alaaeldin El-Nouby, Zhuang Liu, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. ImageBind: One Embedding Space To Bind Them All, 2023. CVPR 2023.
- [25] Siavash Golkar, Jake Kovalic, Irina Espejo Morales, Samuel Sledzieski, Minhuan Li, Ksenia Sokolova, Geraud Krawezik, Alberto Bietti, Claudia Skok Gibbs, Roman Klypa, Shengwei Xiong, Francois Lanusse, Liam Parker, Kyunghyun Cho, Miles Cranmer, Tom Hehir, Michael McCabe, Lucas Meyer, Rudy Morel, Payel Mukhopadhyay, Mariel Pettee, Helen Qu, Jeff Shen, David Fouhey, Hadi Sotoudeh, Vikram Mulligan, Pilar Cossio, Sonya M. Hanson, Alisha N. Jones, Olga G. Troyanskaya, and Shirley Ho. MIMIC: A Generative Multimodal Foundation Model for Biomolecules, 2026.
- [26] Siavash Golkar, Mariel Pettee, Michael Eickenberg, Alberto Bietti, Miles Cranmer, Geraud Krawezik, Francois Lanusse, Michael McCabe, Ruben Ohana, Liam Parker, Bruno Régaldou-Saint Blancard, Tiberiu Tesileanu, Kyunghyun Cho, and Shirley Ho. xVal: A continuous numerical tokenization for scientific language models, 2024.
- [27] Jiaming Han, Kaixiong Gong, Yiyuan Zhang, Jiaqi Wang, Kaipeng Zhang, Dahua Lin, Yu Qiao, Peng Gao, and Xiangyu Yue. OneLLM: One Framework to Align All Modalities with Language, 2023. CVPR 2024.
- [28] Jiaming Han, Renrui Zhang, Wenqi Shao, Peng Gao, Peng Xu, Han Xiao, Kaipeng Zhang, Chris Liu, Song Wen, Ziyu Guo, Xudong Lu, Shuai Ren, Yafei Wen, Xiaoxin Chen, Xiangyu Yue, Hongsheng Li, and Yu Qiao. ImageBind-LLM: Multi-modality Instruction Tuning, 2023.
- [29] Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space, 2024.
- [30] Yaru Hao, Haoyu Song, Li Dong, Shaohan Huang, Zewen Chi, Wenhui Wang, Shuming Ma, and Furu Wei. Language Models are General-Purpose Interfaces, 2022. MetaLM.
- [31] Kexin Huang, Serena Zhang, Hanchen Wang, Yuanhao Qu, Yingzhou Lu, Yusuf Roohani, Ryan Li, Lin Qiu, Gavin Li, Junze Zhang, Di Yin, Shruti Marwaha, Jennefer N. Carter, Xin Zhou, Matthew Wheeler, Jonathan A. Bernstein, Mengdi Wang, Peng He, Jingtian Zhou, Michael Snyder, Le Cong, Aviv Regev, and Jure Leskovec. Biomni: A General-Purpose Biomedical AI Agent. bioRxiv preprint, 2025.
- [32] Xiaohong Ji, Zhen Wang, Zhifeng Gao, Hang Zheng, Linfeng Zhang, Guolin Ke, and Weinan E. Uni-Mol2: Exploring Molecular Pretraining Model at Scale, 2024.
- [33] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-LLM: Time Series Forecasting by Reprogramming Large Language Models, 2023. ICLR 2024.

- [34] Ilay Kamai, Marc-Huertas Company, Mike J. Smith, and Hagai B. Perets. Talking with the Latents – how to convert your LLM into an astronomer, 2026.
- [35] Oğuzhan Fatih Kar, Alessio Tonioni, Petra Poklukar, Achin Kulshrestha, Amir Zamir, and Federico Tombari. BRAVE: Broadening the visual encoding of vision-language models, 2024. ECCV 2024.
- [36] Dongki Kim, Wonbin Lee, and Sung Ju Hwang. Mol-LLaMA: Towards General Understanding of Molecules in Large Molecular Language Model, 2025. NeurIPS 2025.
- [37] Kimi Team. Kimi K2: Open Agentic Intelligence, 2025.
- [38] Daniel Levine, Syed Asad Rizvi, Sacha Lévy, Nazreen Pallikkavaliyaveetil, David Zhang, Xingyu Chen, Sina Ghadermarzi, Ruiming Wu, Ziheng Zheng, Ivan Vrkic, Anna Zhong, Daphne Raskin, Insu Han, Antonio Henrique De Oliveira Fonseca, Josue Ortega Caro, Amin Karbasi, Rahul Madhav Dhodapkar, and David van Dijk. Cell2Sentence: Teaching Large Language Models the Language of Biology. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *PMLR*, pages 27299–27325, 2024. Preprint doi:10.1101/2023.09.11.557287.
- [39] Bangzheng Li, Ximeng Sun, Jiang Liu, Ze Wang, Jialian Wu, Xiaodong Yu, Hao Chen, Emad Barsoum, Muhao Chen, and Zicheng Liu. Latent visual reasoning, 2025.
- [40] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models, 2023.
- [41] Sihang Li, Zhiyuan Liu, Yanchen Luo, Xiang Wang, Xiangnan He, Kenji Kawaguchi, Tat-Seng Chua, and Qi Tian. Towards 3D Molecule-Text Interpretation in Language Models, 2024.
- [42] Zihao Li, Jiaru Zou, Feihao Fang, Xuying Ning, Mengting Ai, Tianxin Wei, Sirui Chen, Xiyuan Yang, and Jingrui He. Heterogeneous Scientific Foundation Model Collaboration, 2026.
- [43] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual Instruction Tuning, 2023. NeurIPS 2023.
- [44] Nuowei Liu, Changzhi Sun, Tao Ji, Junfeng Tian, Jianxin Tang, Yuanbin Wu, and Man Lan. EvoLlama: Enhancing LLMs’ Understanding of Proteins via Multimodal Structure and Sequence Representations, 2024.
- [45] Xiaoze Liu, Ruowang Zhang, Weichen Yu, Siheng Xiong, Liu He, Feijie Wu, Hoin Jung, Matt Fredrikson, Xiaoqian Wang, and Jing Gao. The Vision Wormhole: Latent-Space Communication in Heterogeneous Multi-Agent Systems, 2026.
- [46] Yingzhuo Liu. Beyond tokens: a unified framework for latent communication in LLM-based multi-agent systems, 2026.
- [47] Zhiyuan Liu, Sihang Li, Yanchen Luo, Hao Fei, Yixin Cao, Kenji Kawaguchi, Xiang Wang, and Tat-Seng Chua. MolCA: Molecular Graph-Language Modeling with Cross-Modal Projector and Uni-Modal Adapter, 2023. EMNLP 2023.
- [48] Zhiyuan Liu, An Zhang, Hao Fei, Enzhi Zhang, Xiang Wang, Kenji Kawaguchi, and Tat-Seng Chua. ProtT3: Protein-to-Text Generation for Text-based Protein Understanding, 2024. ACL 2024.

- [49] Yizhen Luo, Jiahuan Zhang, Siqi Fan, Kai Yang, Yushuai Wu, Mu Qiao, and Zaiqing Nie. BioMedGPT: Open Multimodal Generative Pre-trained Transformer for BioMedicine, 2023.
- [50] Valentino Maiorca, Luca Moschella, Antonio Norelli, Marco Fumero, Francesco Locatello, and Emanuele Rodolà. Latent Space Translation via Semantic Alignment, 2023. NeurIPS 2023.
- [51] Adnan Malik, Muhammad Arsalan, Carlos Moreno, Juan Mosquera, Eloy Félix, Tevfik Kizilören, Venkatesh Muthukrishnan, Barbara Zdrazil, Andrew R. Leach, and Noel M. O’Boyle. ChEBI: re-engineered for a sustainable future. *Nucleic Acids Research*, 54(D1):D1768–D1778, 2026.
- [52] Lingchen Meng, Jianwei Yang, Rui Tian, Xiyang Dai, Zuxuan Wu, Jianfeng Gao, and Yu-Gang Jiang. DeepStack: Deeply Stacking Visual Tokens is Surprisingly Simple and Effective for LMMs, 2024. NeurIPS 2024.
- [53] Moonshot AI. Kimi K2.6 model card. <https://huggingface.co/moonshotai/Kimi-K2.6>, 2026. Model Summary; accessed September 29, 2026.
- [54] Luca Moschella, Valentino Maiorca, Marco Fumero, Antonio Norelli, Francesco Locatello, and Emanuele Rodolà. Relative representations enable zero-shot latent space communication, 2022. ICLR 2023.
- [55] Artemis Panagopoulou, Le Xue, Ning Yu, Junnan Li, Dongxu Li, Shafiq Joty, Ran Xu, Silvio Savarese, Caiming Xiong, and Juan Carlos Niebles. X-InstructBLIP: A Framework for aligning X-Modal instruction-aware representations to LLMs and Emergent Cross-modal Reasoning, 2023. ECCV 2024.
- [56] Qizhi Pei, Wei Zhang, Jinhua Zhu, Kehan Wu, Kaiyuan Gao, Lijun Wu, Yingce Xia, and Rui Yan. BioT5: Enriching Cross-modal Integration in Biology with Chemical Knowledge and Natural Language Associations, 2023. EMNLP 2023.
- [57] Bryan Perozzi, Bahare Fatemi, Dustin Zelle, Anton Tsitsulin, Mehran Kazemi, Rami Al-Rfou, and Jonathan Halcrow. Let Your Graph Do the Talking: Encoding Structured Data for LLMs, 2024.
- [58] Chau Pham, Boyi Liu, Yingxiang Yang, Zhengyu Chen, Tianyi Liu, Jianbo Yuan, Bryan A. Plummer, Zhaoran Wang, and Hongxia Yang. Let Models Speak Ciphers: Multiagent Debate through Embeddings, 2023. ICLR 2024.
- [59] Vignav Ramesh and Kenneth Li. Communicating Activations Between Language Model Agents, 2025. ICML 2025.
- [60] Syed Asad Rizvi, Daniel Levine, Aakash Patel, Shiyang Zhang, Eric Wang, Curtis Jamison Perry, Ivan Vrkic, Nicole Mayerli Constante, Zirui Fu, Sizhuang He, David Zhang, Cerise Tang, Zhuoyang Lyu, Rayyan Darji, Chang Li, Emily Sun, David Jeong, Lawrence Zhao, Jennifer Kwan, David Braun, Brian Hafler, Hattie Chung, Rahul M. Dhodapkar, Paul Jaeger, Bryan Perozzi, Jeffrey Ishizuka, Shekoofeh Azizi, and David van Dijk. Scaling Large Language Models for Next-Generation Single-Cell Analysis. bioRxiv preprint, 2025. C2S-Scale.
- [61] Jerret Ross, Brian Belgodere, Vijil Chenthamarakshan, Inkit Padhi, Youssef Mroueh, and Payel Das. Large-scale chemical language representations capture molecular structure and properties. *Nature Machine Intelligence*, 4(12):1256–1264, 2022. MoLFormer.

- [62] Maximillian Rossi, Prajwal Raghunath, and Eugene Wu. Latent Cache Flow: Model-to-Model Communication Without Text, 2026.
- [63] Moritz Schaefer, Peter Peneder, Daniel Malzl, Salvo Danilo Lombardo, Mihaela Peycheva, Jake Burton, Anna Hakobyan, Varun Sharma, Thomas Krausgruber, Celine Sin, Jörg Menche, Eleni M. Tomazou, and Christoph Bock. Multimodal learning enables chat-based exploration of single-cell data. *Nature Biotechnology*, 2025. CellWhisperer; preprint doi:10.1101/2024.10.15.618501.
- [64] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language Models Can Teach Themselves to Use Tools, 2023.
- [65] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face, 2023. NeurIPS 2023.
- [66] Martin Steinegger and Johannes Söding. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature Biotechnology*, 35(11):1026–1028, 2017.
- [67] Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601. Association for Computational Linguistics, 2019.
- [68] The UniProt Consortium. UniProt: the Universal Protein Knowledgebase in 2025. *Nucleic Acids Research*, 53(D1):D609–D617, 2025.
- [69] Maria Tsimpoukelli, Jacob Menick, Serkan Cabi, S. M. Ali Eslami, Oriol Vinyals, and Felix Hill. Multimodal Few-Shot Learning with Frozen Language Models, 2021. NeurIPS 2021.
- [70] Ching-Huei Tsou, Michal Ozery-Flato, Ella Barkan, Diwakar Mahajan, and Ben Shapira. BioVERSE: Representation Alignment of Biomedical Modalities to LLMs for Multi-Modal Reasoning, 2025.
- [71] Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering Language Models With Activation Engineering, 2023. ActAdd; earlier title: Activation Addition: Steering Language Models Without Optimization.
- [72] Chao Wang, Hehe Fan, Ruijie Quan, and Yi Yang. ProtChatGPT: Towards Understanding Proteins with Large Language Models, 2024.
- [73] Eric Wang, Samuel Schmidgall, Paul F. Jaeger, Fan Zhang, Rory Pilgrim, Yossi Matias, Joelle Barral, David Fleet, and Shekoofeh Azizi. TxGemma: Efficient and Agentic LLMs for Therapeutics, 2025.
- [74] Zeyuan Wang, Qiang Zhang, Keyan Ding, Ming Qin, Xiang Zhuang, Xiaotong Li, and Huajun Chen. InstructProtein: Aligning Human and Protein Language via Knowledge Instruction, 2023.
- [75] Zifeng Wang, Zichen Wang, Balasubramaniam Srinivasan, Vassilis N. Ioannidis, Huzefa Rangwala, and Rishita Anubhai. BioBridge: Bridging Biomedical Foundation Models via Knowledge Graphs, 2023. ICLR 2024.

- [76] Yingce Xia, Peiran Jin, Shufang Xie, Liang He, Chuan Cao, Renqian Luo, Guoqing Liu, Yue Wang, Zequn Liu, Yuan-Jyue Chen, et al. Nature Language Model: Deciphering the Language of Nature for Scientific Discovery, 2025.
- [77] Yijia Xiao, Edward Sun, Yiqiao Jin, Qifan Wang, and Wei Wang. ProteinGPT: Multimodal LLM for Protein Property Prediction and Structure Understanding, 2024. MLGenX Workshop at ICLR 2025.
- [78] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models, 2022. ICLR 2023.
- [79] Andy Zeng, Maria Attarian, Brian Ichter, Krzysztof Choromanski, Adrian Wong, Stefan Welker, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, Johnny Lee, Vincent Vanhoucke, and Pete Florence. Socratic Models: Composing Zero-Shot Multimodal Reasoning with Language, 2022. ICLR 2023.
- [80] Yi-Kai Zhang, Shiyin Lu, Yang Li, Yanqing Ma, Qing-Guo Chen, Zhao Xu, Weihua Luo, Kaifu Zhang, De-Chuan Zhan, and Han-Jia Ye. Wings: Learning Multimodal LLMs without Text-only Forgetting, 2024. NeurIPS 2024.
- [81] Zijia Zhao, Longteng Guo, Tongtian Yue, Sihan Chen, Shuai Shao, Xinxin Zhu, Zehuan Yuan, and Jing Liu. ChatBridge: Bridging Modalities with Large Language Model as a Language Catalyst, 2023.
- [82] Xibin Zhou, Chenchen Han, Yingqi Zhang, Huan Du, Jiayuan Tian, Jin Su, Renju Liu, Kai Zhuang, Shiyu Jiang, Anthony Gitter, Li Liu, Huayu Li, Meiqi Wu, Shiyang You, Zichen Yuan, Feng Ju, Huilin Zhang, Wei Zheng, Fengyuan Dai, Yuyang Zhou, Yuyang Tao, Dan Wu, Zongze Shao, Yang Liu, Hongyuan Lu, and Fajie Yuan. Decoding the Molecular Language of Proteins with Evolla. bioRxiv preprint, 2025.
- [83] Yufan Zhuang, Chandan Singh, Liyuan Liu, Jingbo Shang, and Jianfeng Gao. Vector-ICL: In-context Learning with Continuous Vector Representations, 2024.
- [84] Le Zhuo, Zewen Chi, Minghao Xu, Heyan Huang, Heqi Zheng, Conghui He, Xian-Ling Mao, and Wentao Zhang. ProtLLM: An Interleaved Protein-Language LLM with Protein-as-Word Pre-Training, 2024.
- [85] Jiaru Zou, Ruizhong Qiu, Gaotang Li, Xiyuan Yang, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, James Zou, Mengdi Wang, and Ling Yang. Latent Collaboration in Multi-Agent Systems, 2025. ICML 2026.

Supplementary material

A	Methods: building a connector	30
A.1	Models and representations	30
A.2	Data, splits and supervised targets	31
A.3	Question examples and prompt construction	32
A.4	Align and Select	33
A.5	Train	36
A.6	Applying and combining connectors	37
B	Evaluation methods and supporting evidence	37
B.1	Evaluation protocol	37
B.2	Within-task generalization	38
B.3	Transfer to new questions	38
B.4	Effect of training	39
B.5	Merging representations of one protein	39
B.6	Protein-compound relations	40
B.7	Information beyond names	40

A Methods: building a connector

This section gives the construction recipe used for our demonstration. The choices of representation, alignment reference, injection layers and training objective are implementation choices within LIP. The shared evaluation protocol is in Section B.1; additional controls are in Section B.

A.1 Models and representations

We connect each specialist independently to Gemma 3 4B, Gemma 3 12B and Kimi K2.6. All pre-trained weights are frozen, and specialist embeddings are extracted before connector fitting. Table 1 specifies the selected layers, pooling and concatenation order; the same specialist representation is used for all three LLMs.

For proteins, we use MIMIC, ESM-C 600M and PGLM 1B. The ESM-C and PGLM revisions are `fcc9d36f` and `676c1bcb`, respectively. Mean pooling excludes special and padding tokens. MIMIC accepts sequences up to 10,000 amino acids. Longer inputs to ESM-C and PGLM are split into windows of at most 2046 and 1023 amino acids, respectively, and window means are weighted by their lengths.

The molecule models are MoLFormer-c3 (`9f1b9ea3`), ChemFM-1B (`f99dc2e8`) and Uni-Mol2 1.1B through `unimol_tools` 0.1.6. MoLFormer and ChemFM use SMILES, derived from InChI when necessary, truncated to their configured limits of 202 and 512 tokens. Uni-Mol2 uses coordinates with hydrogens removed. Its current extraction retries failed three-dimensional conformer generation from random coordinates, checks specified stereochemistry, and otherwise retains the original fallback. The current Uni-Mol2 extraction uses isolated padding and no atom cropping. These extraction choices are part of the connector’s representation definition.

Table 3: Task supervision and reserved categories. Reserved questions may have related information in description targets.

Domain	Task supervision	Reserved question categories
Protein	Function, family, subunit, catalytic activity, EC class, pathway, location and selected keywords; relations and variant effects, pathogenicity and loss of function.	Lineage, length, selected modifications, cofactors and domains.
Compound	Definitions, mechanism roles, structural classes and functional parents; relations between compounds.	Selected provenance, application and chemical roles; formula, mass, charge and element-based questions.

A.2 Data, splits and supervised targets

A.2.1 Sources and object separation

Protein sequences and annotations come from reviewed UniProtKB/Swiss-Prot entries, release 2026.03, and compounds from three-star ChEBI entries, release 255. Text targets use the corresponding annotation fields with source citations and markup removed. Definitions, classes, roles and functional-parent relations are processed separately to control which information is used for fitting. Rhea supplies reaction relationships for cross-domain evaluation.

Protein splits use MMseqs2 at 30% sequence identity and 80% coverage of both sequences, checked in both search directions at $E \leq 10^{-3}$. Related chemical forms are grouped before molecule splitting, with similarity links at ECFP4 Tanimoto similarity 0.5 (radius 2, 2048 bits). Cross-split links are removed by reassignment or exclusion; only objects with the required embeddings and annotations enter fitting. These exclusions apply to every object in a training prompt, including worked examples and relational partners. They concern connector training; overlap with LLM or specialist pretraining is unknown.

A.2.2 Supervised and reserved targets

Table 3 summarizes the task partition. Names provide alignment references and textual controls, but naming the object is not a training target. Annotation-derived negatives indicate missing annotations, not verified biological or chemical absence. A question withheld from task training may still be answerable from descriptions or correlated annotations (Section B.3.1).

Molecule roles are separated by ontology ancestry before the remaining mechanism roles are assigned to training; class labels include transitive `is_a` ancestry. The partition accounts for related ontology labels as well as the named categories themselves. The mass threshold is fixed at 329.29 Da, the median of the original training split, and retained after subsequent data filtering.

A.2.3 Sampling and relational examples

Protein training draws 40% single-object text, 20% Yes/No, 20% relational and 20% variant examples. Molecule training draws 50% text, 25% Yes/No and 25% relational examples (Figure 3). For ordinary object sampling, an object’s weight is proportional to $1/\sqrt{n_c}$, where n_c is its cluster size. Yes/No sampling balances positive and negative labels after selecting a question type. The variant stream includes mutation-effect descriptions, pathogenicity and loss-of-function questions, and wild-type–mutant comparisons.

Relational answers state annotation-derived facts about one, two or three objects and identify them by prompt position. Half of odd-one-out draws use hard negatives differing at the annotation level requested. Every injected slot within a training example uses the same connector; combining independently trained connectors is reserved for evaluation.

A.3 Question examples and prompt construction

The examples below illustrate the question forms used in our experiments. <SLOT> denotes an injected position. Chat-template tokens and worked-example preambles are omitted; the displayed targets illustrate annotation supervision, not generated responses. Descriptions and binary questions concern one object, relational questions compare objects, and combined representations and cross-domain questions test composition of independently trained connectors.

Protein description.

What does UniProt say about its subunit structure? The protein is<SLOT>.

Target: The complex is composed of two ATP-binding proteins, two transmembrane proteins and a solute-binding protein.

Compound description.

What is the compound given as<SLOT> active as?

Target: Role: antifungal agent.

Yes/No property question.

Here is an unknown protein, given to you as an embedding:<SLOT>. Does this protein bind DNA? Answer Yes or No.

For compounds, the same frame asks, for example, Is this compound a lipid?, Is this compound a metabolite?, or Is the molecular mass of this compound above 329 Da?

Several properties in one question.

Give a short profile of the protein<SLOT>: its family, where it is found and what it does.

Relational comparison.

Do the proteins<SLOT> and<SLOT> belong to the same family? Explain.

A compound example is Do<SLOT> and<SLOT> derive from the same parent compound?

Relational choice.

Three proteins:<SLOT>,<SLOT>,<SLOT>. Two share a family. Which one does not?

Target: The first and third belong to the CDC42SE/SPEC family; the second belongs to the CTF8 family. The second is the odd one out.

Variant effect.

Here is a mutant protein, given to you as an embedding:<SLOT>. What is the effect of its mutation?

Variant pathogenicity and loss of function.

Is the following missense variant pathogenic? Variant:<SLOT>

Consider the protein variant<SLOT>. Does this variant lose activity?

Both use a binary answer convention.

Wild-type-mutant comparison.

Protein 1:<SLOT>. Protein 2:<SLOT>. Protein 2 is a point mutant of protein 1. What is the effect of the mutation?

Several representations of one object.

Here is an unknown protein, given to you as an embedding:<SLOT><SLOT><SLOT>.

Does this protein bind DNA? Answer Yes or No.

The slots contain different specialists' representations of the same protein.

Cross-domain question.

Here is an unknown protein, given to you as an embedding:<SLOT>. Here is an unknown compound, given to you as an embedding:<SLOT>. Does this protein catalyze a reaction in which this compound is a substrate or a product? Answer Yes or No.

Name elicitation.

Here is an unknown protein, given to you as an embedding:<SLOT>. What is the name of this protein? Reply with the protein name only.

The response can replace the embedding in a subsequent property question; a supplied-name control instead uses the database name.

Text and Yes/No training prompts vary wording, slot placement and requested answer format through a compositional grammar. Worked examples use other training objects outside the query’s cluster, and every injected example uses the same connector. Selected phrasings are reserved for evaluation. Text and variant targets are capped at 48 tokens and relational targets at 96; truncation does not add an end-of-answer token. Binary evaluations use a single user turn and the Yes/No frame above, with reasoning disabled unless explicitly tested. Requests for reasoning change the response procedure described in Section 4.4.

A.4 Align and Select

A.4.1 Specialist read profiles

Before alignment, ridge probes compare specialist layers using property labels and five cluster-separated folds. For proteins, additional clinical and functional variant probes use the difference between mutant and wild-type representations. These probes assess the specialist representation without using the LLM. Candidate layers and concatenations are compared by task performance, leading to the human-selected representations in Table 1 and those used in the scaling experiments.

Figures 15 and 16 show layer profiles alongside the chosen representations. Property means use the per-specialist task subsets retained in the original scan summaries, applying the same subset to every representation of that specialist. The subsets differ across specialists, so these curves primarily compare layers within a specialist. Protein variant curves average the clinical and functional probe AUROCs. ChemFM 3B uses all 59 property probes and concatenates the last-token and mean-pooled embeddings at each layer. The profiles are exploratory guidance for representation choice.

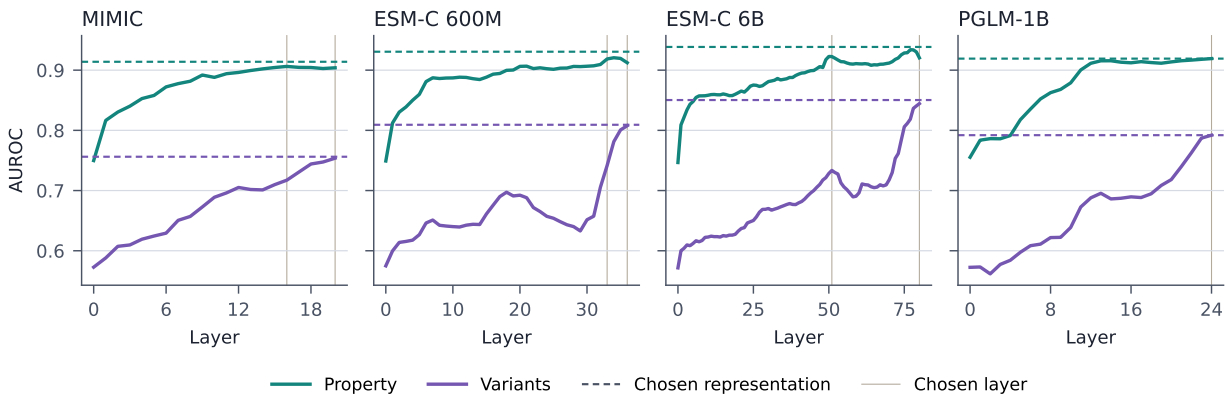


Figure 15: **Protein specialist read profiles.** Mean AUROC of property (green) and variant (purple) probes for mean-pooled representations across specialist layers. Vertical lines mark chosen layers, and dashed lines show the corresponding scores of the chosen single-layer or concatenated representation.

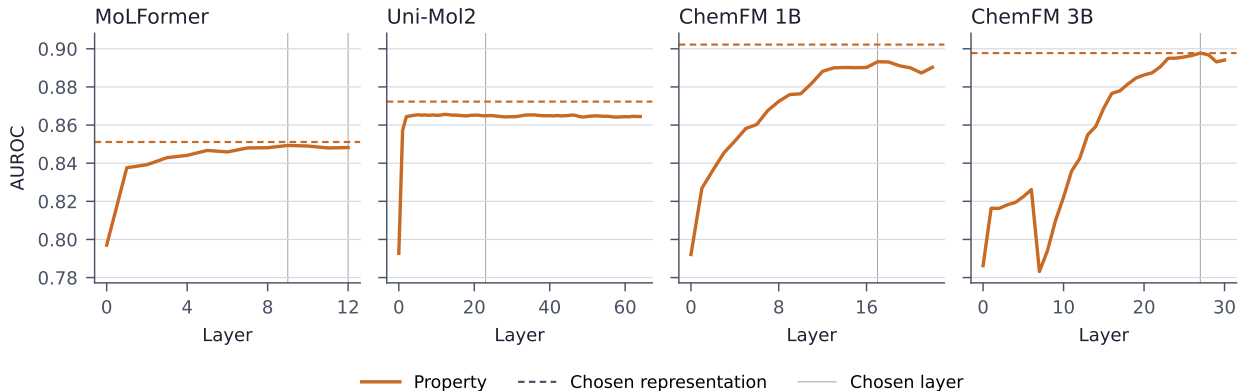


Figure 16: **Molecule specialist read profiles.** Mean property-probe AUROC across specialist layers, using mean-pooled embeddings for MolFormer, Uni-Mol2 and ChemFM 1B, and concatenated last-token and mean-pooled embeddings for ChemFM 3B. Vertical lines mark chosen layers, and dashed lines show the scores of the full chosen representations.

A.4.2 Standardization and the write operation

Coordinate means and standard deviations are fitted on the regression objects and fixed thereafter: $z_j = (e_j - \mu_j)/(s_j + 10^{-8})$. At selected layer ℓ , an affine map $\delta_\ell = W_\ell z + b_\ell$ supplies the slot’s write direction. Normalization and mixing (Equation 1) use float32 with norm denominators bounded below by 10^{-6} before casting back to the LLM’s computation dtype. A learned object-independent constant supplies the input-embedding slot direction. Other token positions undergo ordinary LLM computation.

A.4.3 Reference states and ridge fitting

Reference states are captured in prompts containing an object’s description followed by its name. Protein targets average the name token states; molecule targets use the final name token. For

Kimi, the target for write layer $\ell > 0$ comes from layer $\max(\ell, p_*)$, with $p_* = 13$ for proteins and 9 for molecules; these reference depths were selected using the same diagnostic AUROCs. Gemma targets use the matching layer. This reference-depth rule sets the regression target and does not restrict where the connector may write.

Ridge maps fit centered reference states from standardized specialist representations, with the intercept set by the reference mean. The penalty is selected by five-fold cross-validation over 28 logarithmically spaced values from 10^0 to 10^9 . The scan objects are excluded from both fitting and standardization for selection maps; the production maps use the full fitting set. Reference capture uses names only during construction, not when the connector receives a new object.

A.4.4 Name states as an alignment reference

We compared final name-token states with mean description states from either bare text or an assistant response following a prompt containing the name (Figure 17). Ridge maps used the same 9017 fitting proteins; injected-task performance was averaged over five Yes/No questions on 200 held-out proteins. Name-state predictions gave better injected-question performance. This illustrates why alignment quality is assessed through the LLM’s responses.

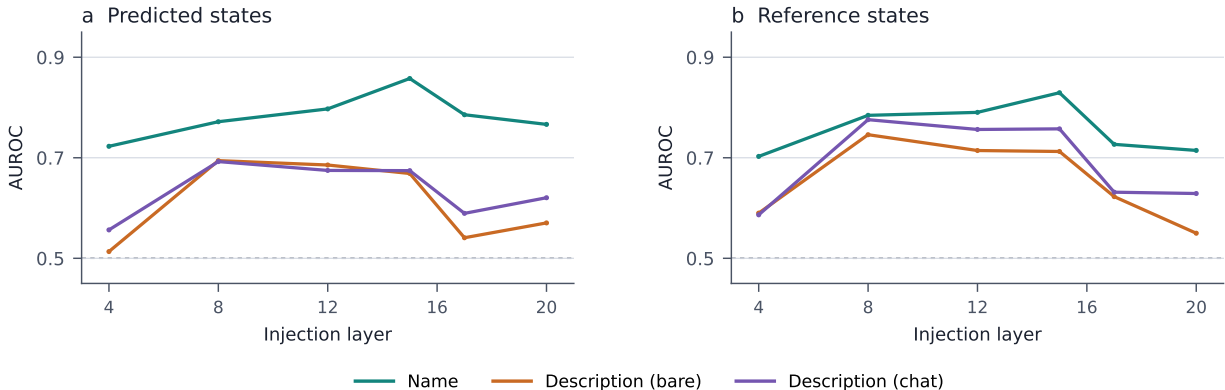


Figure 17: **Name and description states as alignment targets.** Mean task AUROC at each scanned layer in the informative band of Kimi K2.6, injecting (a) ridge-predicted states or (b) the LLM’s actual reference states.

A.4.5 LLM write profiles and selection

Each ridge prediction is injected separately at each LLM layer. Mean AUROC across five diagnostic questions gives a curve for each connector, alongside a curve obtained by injecting the LLM’s own reference states (Figure 18). The questions concern RNA binding, transcription regulation, hydrolase activity, membrane association and Pseudomonadota for proteins, and benzenoid aromatic compounds, esters, lipids, drug roles and net charge for molecules. These questions guide layer selection.

For each domain and LLM, local maxima of the maximum envelope over connector and reference curves guide the choice of layers spanning depth (Table 2). The shared schedule is a choice for this demonstration. The figure shows the three specialists in each domain from Table 1 and the maximum across those curves and the own-state reference. The original Kimi protein selection

also considered an ESM-C 6B curve, which is omitted from this display; the selected layers are unchanged.

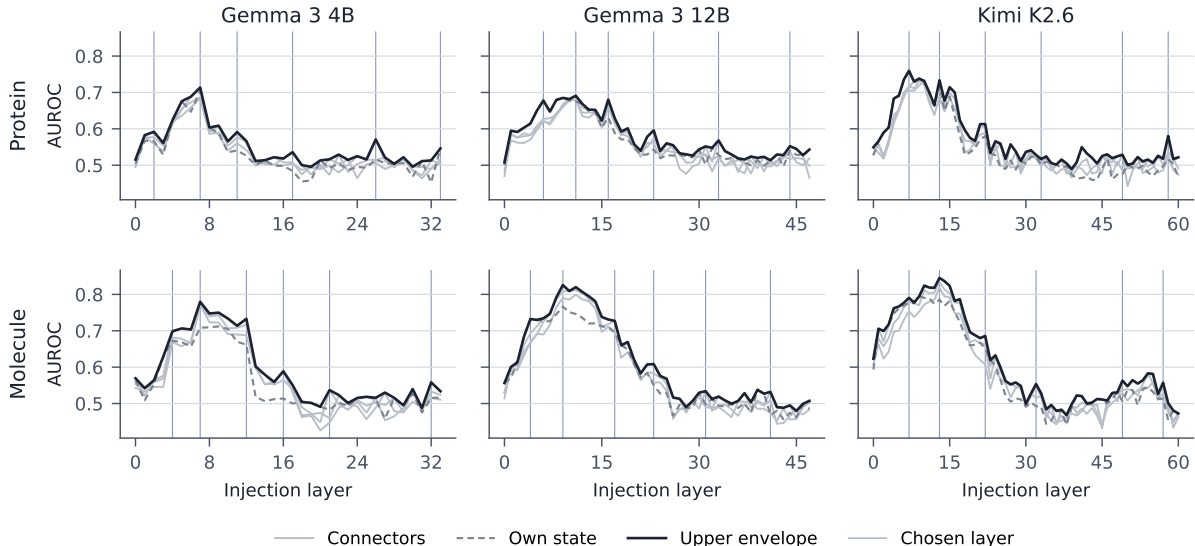


Figure 18: **LLM injection-layer profiles.** Mean diagnostic-task AUROC from single-layer injection of aligned connector outputs (gray) and the LLM’s own reference states (dashed), with the maximum across the displayed curves in black and chosen layers marked vertically. Rows show proteins and molecules; columns show Gemma 3 4B, Gemma 3 12B and Kimi K2.6.

Mixing weights are initialized from each connector’s above-chance mean AUROC on the trained diagnostic questions, normalized by its best layer and clipped to $[0.1, 0.9]$, then learned during training.

A.5 Train

A.5.1 Objective and token weights

Only the connector’s affine maps, biases, input constant and mixing weights are trained. Gradients pass through the frozen LLM from answer tokens to these parameters. With teacher-forced negative log likelihood ℓ_{it} and answer-token weight w_{it} , the objective is

$$\mathcal{J} = \frac{\sum_{i,t} w_{it} \ell_{it}}{\sum_{i,t} w_{it}}. \quad (2)$$

Prompt tokens receive no loss. Position-dependent weights emphasize annotation content whose prediction benefits from knowing the object’s name; connecting text and end tokens receive a floor weight. Each factual span of a relational answer restarts its annotation-specific position weights, and binary decision tokens receive full weight before stream-specific scaling.

A.5.2 Optimization and batching

We use AdamW with a separate base learning rate for each map-weight tensor, bias vector and input constant, set to 0.002 times the root mean square of that tensor’s initial values. Across the 18 connectors, the resulting absolute learning rates range approximately from 3×10^{-7} to

2×10^{-2} for map weights, 8×10^{-5} to 3 for biases, and 4×10^{-6} to 6×10^{-4} for input constants; mixing logits use a learning rate of 0.01. A 20-step linear warmup precedes a constant learning rate; gradients are clipped to global norm 1.0. Per-step relative weight decay of 2.5×10^{-4} applies to the maps, scaled by warmup and independent of learning rate; biases, the input constant and mixing logits are excluded. Each optimizer update targets 85 examples, with microbatches where needed to fit memory. Main comparisons use the final checkpoint after 2,000 updates, without selecting an intermediate checkpoint by evaluation performance. Main experiments use one training seed per connector; the scaling comparison includes the repeated runs described in Supplementary Section B.1.

A.5.3 Training through the frozen LLM

Kimi training adapts the expert-parallel implementation described by Workshop Labs for Trelis [21], retaining INT4 expert weights with bfloat16 computation; Gemma uses a single-GPU bfloat16 backend. Backpropagation computes gradients through the LLM with respect to its inputs, without LLM weight gradients or optimizer states. Packed examples retain separate causal-attention boundaries and restart position indices for each example. Section 3.3 gives representative training costs.

A.6 Applying and combining connectors

Each slot selects a connector and an embedding extracted according to its representation definition. The slot retains its own normalization, maps, write layers and mixing weights; where that connector has no write, the LLM proceeds normally. Different connectors need not share injection schedules. Several slots may encode one object through different specialists or supply different objects, without combining or refitting connector parameters.

A checkpoint identifies its LLM, specialist, extraction procedure, normalization and write schedule. These must match at inference because the maps are specific to the model pair and representation. All inference backends apply the same recorded writes at the specified slots and layers.

B Evaluation methods and supporting evidence

The following methods and controls address whether the LLM uses the supplied object, transfers beyond supervised questions, combines independently trained connectors and accesses information beyond names.

B.1 Evaluation protocol

Objects and tasks. Connector training and evaluation use the object exclusions described in Section A.2. Read and write selection use training-pool objects; write-scan objects are excluded from fitting the selection maps but are not a held-out evaluation population. Monitoring and final evaluations are distinguished from these construction scans. Cross-domain populations additionally distinguish familiar and unseen objects in each domain. Matched comparisons use the same eligible objects and questions for the conditions being compared. Enrichment for uncommon labels changes the evaluated population relative to a random sample.

Input conditions. The *own* condition supplies the queried object’s embedding. A *shuffled* control substitutes a different object’s embedding while retaining the question, target and connector.

The *placeholder* condition leaves the slot uninjected. The *name* condition replaces it with the database name as text; names can activate LLM prior knowledge but are not an upper bound on specialist information. An *initial* connector has completed *Align* and *Select*, before answer-loss training. Question examples appear in Section A.3.

Scores. For binary questions, the score is $s_i = \log p(\text{Yes} \mid x_i) - \log p(\text{No} \mid x_i)$ at the answer position, with AUROC computed against task labels. Group means weight their tasks equally; chance is 0.5. For a supplied reference answer, the information-access gap is

$$\Delta_i = L_i^{\text{shuf}} - L_i^{\text{own}}, \quad (3)$$

where L_i is mean answer-token negative log likelihood. A positive gap means that the correct embedding helps predict that reference; it does not measure the accuracy of generated text. Relational ordering and within-name concordance are defined with their controls in Section B.

Uncertainty. Where intervals are shown, matched conditions share resampled objects, and task averages share the same resample across tasks. Within-name comparisons resample name groups. In Figure 13, Gemma protein property and ChemFM 3B molecule results average three training seeds; their approximate 95% intervals combine object sampling variance with the between-seed variance of the mean. The remaining conditions in that figure use one seed and percentile bootstrap intervals. Variant intervals resample parent protein clusters. Rare labels can leave some resamples without both classes and therefore without a defined AUROC; narrow intervals in such cases do not establish precise performance.

B.2 Within-task generalization

Unseen question wording. Selected question phrasings are withheld during connector training. A matched control asked the same questions about the same proteins using either training or reserved wording. Paired confidence intervals for changes in the information-access gap included zero for each protein connector, with no detected loss under the reserved wording. This controls wording while holding the object, target and context fixed.

Task groups. Displayed groups use fixed task memberships and equal-weight means of task AUROCs. Protein groups collect related properties; molecule groups use chemical meaning and ChEBI ancestry. These are summaries of correlated questions, not independent dimensions of information. Rare-label limitations follow Section B.1.

B.3 Transfer to new questions

B.3.1 Description exposure

Withholding a task question does not remove its answer from all other supervision. We estimate how often a positive object’s description target conveys the queried property, using the training sampler’s object and text-stream weights and counting repeated presentations relative to all training examples. Property cues are matched in capped descriptions: a 200-character approximation for proteins and 48 Kimi tokens for molecules.

Low-exposure question types have estimated exposure below 0.1%; the main figures also identify starred types with exposure up to 0.32%. These estimates concern description targets, not all routes by which information could enter supervision: variant-effect text, relational answers and worked

context are outside this metric, and correlated or implicit clues may remain. Read-layer probes and write-selection diagnostics can also expose labels for questions withheld from connector task training. Thus task transfer does not imply that the complete construction procedure was blind to every tested property.

B.3.2 Tasks and populations

Transfer questions concern protein lineage and properties, compound roles and mass. Molecule transfer uses held-out structural groups without a training analog at ECFP4 Tanimoto similarity 0.5. The mass threshold is fixed at 329.29 Da, the median of the original training split, and retained after subsequent data filtering. Protein length uses 320 amino acids. Group means weight questions equally, with lineage labels nested and correlated. Each comparison is restricted to objects eligible for its tasks and input conditions.

B.4 Effect of training

Figure 7 compares the same connector after *Align* and *Select* and after *Train*, on matched questions and objects. This isolates the change during connector task training while holding the specialist and LLM fixed. It does not isolate individual supervision streams or distinguish which correlated training targets caused a transfer gain or loss.

B.5 Merging representations of one protein

The composition comparison supplies the same protein through adjacent slots from MIMIC, ESM-C and PGLM, without joint fitting. Repeated-source triples (MMM, EEE and PPP) control for the effect of supplying more slots without adding a different specialist. Figure 19 compares the three different specialists with the best single connector and the best repeated-source triple. Each comparator is selected separately for trained and unseen tasks by mean task AUROC and held fixed across paired resamples.



Figure 19: **Complementary specialists improve information merging.** Mean task AUROC with Kimi K2.6 on 2,036 proteins for (a) five trained and (b) eight unseen questions, comparing three different specialists with the best single connector and the best triple repeating one specialist. Intervals are 95% paired protein-bootstrap intervals (2,000 draws).

These composition comparisons are exploratory. A separate prespecified test required improvement over both controls on at least two of three metrics, with paired intervals above zero; that criterion was not met.

B.6 Protein–compound relations

Tasks and populations. Rhea reaction participation asks whether a protein catalyzes a reaction in which the compound is a substrate or product. BindingDB supplies protein–compound binding questions. Relations stated in training descriptions are excluded, and populations distinguish whether each object appeared in its own connector’s training pool. The Rhea comparison in Figure 8c uses unseen proteins with familiar or unseen compounds. The connectors are fitted independently, without cross-domain question supervision. Annotation-negative pairs indicate absent annotations rather than verified biochemical non-participation. These population comparisons are exploratory; differences between them do not isolate the causal effect of training exposure.

Matched groups and ordering. Each group starts from two annotated protein–compound pairs; swapping compounds yields two mismatched pairs made from the same objects. Each annotated pair is compared with each mismatched pair sharing its protein or compound, using the Yes/No log-probability contrast. Correct orderings score one, ties one half and incorrect orderings zero; the four comparisons are averaged within a group, then across groups. A score additive in separate protein and compound contributions averages exactly 0.5, so above-chance ordering requires a relationship between inputs.

B.7 Information beyond names

Replacing embeddings with elicited names. The LLM first generates a name from the protein embedding. That response then replaces the slot in the same property questions. Every eligible object remains, whether or not its generated name is correct. Comparisons use matched objects and question sets and average task AUROCs equally. A decline after replacement can reflect an incorrect name as well as information lost by compressing the embedding to a name.

Identical-name groups. Objects are grouped by the exact supplied database-name string. Within-name concordance compares positive and negative objects only within such groups, counting ties as one half and pooling pairs across groups for each question. Question-level concordances are then averaged equally. A deterministic supplied-name score is identical for a fixed question within each group and therefore scores 0.5. Above-chance concordance establishes that the embedding provides distinctions unavailable from that shared name alone. Figure 10b–c applies this test to four low-exposure lineage questions and to protein length above 320 amino acids. Only groups containing both labels are informative; nested lineage labels and limited informative groups constrain the interpretation. Uncertainty resamples name groups rather than treating all within-group pairs as independent.